

LEUPHANA
UNIVERSITÄT LÜNEBURG

FInAL
Forum Informatics At Leuphana

20. Jahrgang, Heft 2, April 2010, ISSN 0939-8821

Stimulating Computing

Praxisnahe Wirtschaftsinformatik in Lehre und Forschung

Festschrift für
Prof. Dr. rer. publ., Dipl.-Ing., Dipl.-Wirtsch.-Ing.
Hinrich E. G. Bonin

01000110 01001001 01101110 01000001 01001100

Information Science: Technical Reports and Working Papers
Hrsg: Horst Meyer-Wachsmuth
Volgershall 1, D-21339 Lüneburg

Inhaltsverzeichnis



Prof. Dr. rer. publ., Dipl.-Ing., Dipl.-Wirtsch.-Ing.
Hinrich E. G. Bonin
bonin@uni.leuphana.de

Jahrgang:	1945
Ausbildung:	
∗ Abitur	1964
∗ Hochschulabschlüsse	1970 Dipl.-Ing. Elektrotechnik (Nachrichtentechnik), Technische Hochschule Darmstadt 1973 Dipl.-Wirtsch.-Ing. (Arbeits- und Wirtschaftswissenschaften (Datenverarbeitung), Technische Universität München
∗ Promotion	1985 Dr. rer.publ. (Verwaltungswissenschaften, Deutsche Hochschule für Verwaltungswissenschaften Speyer)
Berufliche Entwicklung:	
∗ 1964 - 1970	Praktikant bei der Deutschen Bundesbahn, Akermans (Schweden) und der Siemens AG München
∗ 1970 - 1971	Zentrallaboratorium für Datentechnik der Siemens AG München
∗ 1973 - 1975	Verantwortlicher Systemdesigner für Entwicklung und Implementation im Bereich "Haushalts-, Kassen- und Rechnungswesen", HIS GmbH Hannover
∗ 1975 - 1985	Wissenschaftlicher Angestellter (BAT Ia), Leiter des Bereichs "Administrative Datenverarbeitung der Datenverarbeitungszentrale der Wasser- und Schifffahrtsverwaltung des Bundes (DVZ), ab 1978 stellvertretender Leiter der DVZ
∗ 1985 - Sep.1988	Professor (C2) im Studiengang Systemanalyse der Hochschule Bremerhaven
∗ seit Okt.1988	Professor (C3) im Fachbereich Wirtschaft der Fachhochschule Nordostniedersachsen bzw. Universität Lüneburg für IT-Anwendungen in der öffentlichen Verwaltung während dieser Zeit unter anderem tätig: - in der Selbstverwaltung o Senat, Mitglied und stellvertretendes Mitglied o Konzil, Mitglied o IT-Steuerung o IT-Beauftragter o Sportbeauftragter - Mitwirkungen in der Gesellschaft für Informatik e.V. - Fachliche Leitung der GI-FB6-Tagung und von 4 Tagungen zu praxisrelevanten Themen beim Einsatz von SAP-Software (R/2 & R/3) - Herausgeber der Reihe FInAL
Hobbys:	Kommunalpolitik, Sport, Jagd

Vorwort

H. Meyer-Wachsmuth

Bei der Wahl des Titels dieser Festschrift „*Stimulating Computing*“, die wir zum Ausscheiden unseres Kollegen Prof. Dr. rer. publ., Dipl.-Ing., Dipl.-Wirtsch.-Ing. Hinrich E.G. Bonin hiermit vorlegen, haben wir uns von Bonin selbst inspirieren lassen, nämlich durch Titel eigener Veröffentlichungen[1], die verdeutlichen, wie anregend, stimulierend er seine Disziplin – die Wirtschaftsinformatik - empfunden und mit wie viel Freude er sein Fach gelehrt hat.

Wir, seine Kollegen, Mitarbeiter und Studierende[2], haben selbst erlebt, wie Bonin die Verbreitung und Vertiefung der Nutzung von Computern in der Hochschule und bei den Studierenden stimuliert und gefördert hat.

Der interdisziplinäre Ansatz der Wirtschaftsinformatik kam dem vielseitigen Hochschullehrer Bonin entgegen; hatte er doch Nachrichtentechnik, Arbeits- und Wirtschaftswissenschaften (Datenverarbeitung) und Verwaltungswissenschaften gelernt, praxisnah angewendet und in diesen Gebieten vielfältig[3] publiziert.

Diese Vielfalt spiegelt sich wider in der thematischen Breite der Beiträge in unserer Festschrift.

Die SAP- basierte „Entwicklung einer Fallstudie für die Lehre im IT-gestützten Personalmanagement“ eröffnet die Reihe der Fachartikel ganz im Sinne Bonins, der über Jahre den exemplarischen Einsatz von SAP in der Lehre gefördert hatte.

Nicht unmittelbar zur Lehre in der Wirtschaftsinformatik (WI)[4] gehört das Compilerbauthema des nächsten Beitrages in diesem Heft „Closures, Continuations und

---

[1] Bonin, H. E. G.; „Faszination Programmierung“; FINAL 16. Jg. Heft 2, August 2006, ISSN 0939-8821

Bonin, H. E. G.; „The Joy of Computing“; FINAL 3. Jg. Heft 5, September 1993, (unvollständige Vorabfassung vom Juli 1994) ISSN 0939-8821

[2] Natürlich sind in den vorliegenden Artikeln immer weibliche und männliche Vertreter aller erwähnten Personengruppen angesprochen. Aus Gründen der Lesbarkeit wird auf Kunstwörter wie z. B. StudentInnen verzichtet .

[3] ca. 30 Veröffentlichungen, darunter vier Bücher; Quelle: *Personal Record Bonin* (http://as.uni-lueneburg.de/lebenslauf.html abgerufen am 14.1.10)

[4] Bonin, H. E. G.; “Compilerbau in der Wirtschaftsinformatik” in „Facettenreiche Wirtschaftsnformatik“; FINAL 19. Jg. Heft 1, August 2009, ISSN 0939-8821

die Java Virtual Machine“. Der Compilerbau gehört jedoch – wie Bonin argumentiert[5] – zu einer bildungsorientierten Lehre der WI und bleibt ein interessantes Forschungsfeld der Informatik. Dieses wird durch ständig neu entstehende Programmiersprachen dokumentiert[6]. Softwareingenieure entwickeln offenbar immer neue Ideen, wie professionelle Programmierung leichter und sicherer werden kann.

Im Gegensatz zum professionellen Softwareengineering steht die zunehmende Softwareentwicklung durch *Enduser* in der Praxis. Der Beitrag „Methoden der Spreadsheet-Entwicklung“ thematisiert das grundsätzliche Problem, „dass fehlerhafte Spreadsheets keine Seltenheit sind“, wenn sie denn ausreichend komplex sind. Der leidenschaftliche Softwareentwickler Bonin hat immer dafür gestritten, seinen Studierenden eine gründliche Ausbildung im Engineering komplexer Systeme bieten zu können. Dieses natürlich nicht mit EXCEL, sondern stets mit den modernsten Programmiersprachen[7]. Die er immer schnell aufgegriffen und gelehrt hat.

Komplexe Softwaresysteme erfordern ingenieurmäßiges Modellieren von Daten auf wissenschaftlicher Grundlage. So spiegelt der Beitrag über „Methoden zur Untersuchung komplexer Datenmodelle“ den wissenschaftlichen Anspruch des (Software)Ingenieurs Bonin, den er jederzeit auch in seinen Vorlesungen aufrecht erhalten hat – manchmal zum Leidwesen seiner Studierenden.

Für den Wirtschaftsingenieur Bonin und seine Studierenden der Wirtschaftsinformatik (WI) sind betriebswirtschaftliche Problemstellungen Ausgangspunkt ihrer beruflichen Tätigkeiten, die in betriebliche Strategien eingepasst sein müssen (*IT-Governance*). Der Beitrag „Reflexionen über eine strategiegetriebene, adaptive Budgetierung“ zeigt eine betriebswirtschaftliche Facette eines Studiengangs der WI und eine von Wirtschaftinformatikern zu automatisierende Problemstellung bzw. -lösung[8].

Es war unser Wunsch, durch Beiträge über studentische Projekte in der vorliegenden Festschrift hervorzuheben, welche stimulierenden und für die Lehre innovativen Projekte von Bonin initiiert und geleitet wurden. Die beiden Beiträge „Praxisorientierte und innovative Lehre in der Wirtschaftsinformatik“ und „Von der Massendatenerfassung zur Klimadatenbank - Projektarbeiten im Fach Anwendungsentwicklung 2“ berichten davon,

---

[5] Bonin, H. E. G.; “Compilerbau in der Wirtschaftsinformatik” in „Facettenreiche Wirtschaftsnformatik“; FINAL 19. Jg. Heft 1, August 2009, ISSN 0939-8821.

[6] seit dem Jahr 2000 sind laut http://de.wikipedia.org/wiki/Zeittafel_der_Programmiersprachen (abgerufen am 11.2.10 mindestens 15 neue Programmiersprachen(varianten) entwickelt worden.

[7] z. B. Bonin, H.: Software.Konstruktion mit LISP, de Gruyter, Berlin u.a. 1991, ISBN 3-11-011786-X

Bonin, H.: Der Java-Coach --- Modellieren mit UML, Programmieren mit JDK, Dokumentieren mit HTML4.0 --- (wesentliche zweite Überarbeitung Dec-2002, erste wesentliche Überarbeitung Apr-2001, Original 1998)

[8] Bonin, H. E. G.; “Controlling - Realisieren von Computernutzen” in „Facettenreiche Wirtschaftsnformatik“; FINAL 19. Jg. Heft 1, August 2009, ISSN 0939-8821

wie Studierende sich (die ECTS-Creditpoint-Workload überziehend) begeistern lassen und engagiert in die Tiefen technischer Probleme eintauchen, sie lösen und dabei auch noch organisatorische Hindernisse sowie zwischenmenschliche Friktionen überwinden. Ohne eine solche Erfahrung wollte Bonin seine Studierenden nicht in die Praxis entlassen. Er hat damit einen wesentlichen Beitrag zum guten Ruf unserer Institution geleistet.

Die Abwechslung in den Themen lässt uns hoffen, dass nach dem Motto „Wer vieles bringt, wird jedem etwas bringen“ (Goethe, Faust) viele Leser Anregendes finden.

Wir wünschen viel Freude bei der Lektüre und Erfolg „beim Vermehren der Einsichten“!

Grußwort des Dekans

Lieber Herr Kollege Prof. Dr. rer. publ., Dipl.-Ing., Dipl.-Wirtsch.-Ing. Bonin,
lieber Hinrich,

Du wurdest im Oktober 1988 an den Fachbereich Wirtschaft der Fachhochschule Nordostniedersachsen berufen. Du kamst von der Hochschule Bremerhaven, an der ich ein paar Jahre zuvor als Lehrbeauftragter tätig war. Daher war mir klar: wir hatten eine gute Wahl getroffen.

Dein Lehrgebiet „DV-Anwendung in der öffentlichen Verwaltung" war eine echte Bereicherung und Ausweitung des Lehrangebots unserer Hochschule. Mit großem Elan und Kreativität hast Du Deine Fachdisziplin und Deine Forschungs- und Lehrinteressen in die Wirtschaftsinformatik und den Fachbereich eingebracht. Als (Mit-) Herausgeber und Autor der Reihe „Programmierung komplexer Systeme" im de Gruyter Verlag und als Herausgeber der Reihe FINAL (Fachhochschule Nordostniedersachsen Informatik Arbeitsberichte Lüneburg, später Forum Informatics at Leuphana) hast Du Deine Ideen nicht nur einem breiten Fachpublikum vorgestellt, sondern auch Deinen Fachkollegen Anregung und eine hervorragende Gelegenheit geboten, ihre Ideen zu publizieren. Dein Engagement und deine Beharrlichkeit haben dazu beigetragen, das fachliche Renommee der Wirtschaftsinformatik nicht nur an dieser Hochschule zu fördern.

Wie Studierende berichten, waren Deine gelegentlich unkonventionellen Lehrmethoden manchmal etwas gewöhnungsbedürftig, aber inhaltlich immer anspruchsvoll und anregend, besonders für die uns sehr lieben fachlich interessierten Studierenden. Und das zeichnet einen guten Hochschullehrer doch aus.

Daneben hast Du mit großem Engagement an der Organisation des Fachbereichs und der Hochschule mitgewirkt. So hast Du unsere Interessen im Senat und im Konzil (ein Gremium, dessen Bedeutung kaum noch jemand kennt) vertreten. Besonders aber wirst Du uns als Begründer und langjähriger Leiter des IT-Teams in Erinnerung bleiben. Ohne Deine Mitwirkung hätten wir in diesem für den reibungslosen täglichen Betriebsablauf wichtigen Bereich keine fachlich so gut qualifizierte und kollegiale Mannschaft.

Natürlich werden wir die anregenden Gespräche in der „Lobby der Fakultät", beim Kaffeetrinken, vermissen. Hier gabst Du Deinen Kollegen nicht nur Einblicke in die Denkweisen der kommunalen und überregionalen Politik. Hier wurden nicht selten wichtige Maßnahmen, den Fachbereich und das Department Wirtschaftsinformatik betreffend, diskutiert und vorbereitet.

Über Deine weiteren vielfältigen Interessen vom Hochleistungssport bis zur Jagd gibt Deine Homepage Auskunft. Nebenbei: Hier findet man auch die Bedeutung von „E.G." in Deinem Namen.

Lieber Hinrich, wenn Du jetzt in Deinen wohlverdienten Ruhestand wechselst, wirst Du bei uns eine Lücke hinterlassen. Nicht nur, dass wir über die Organisation und Ausrichtung des Instituts für Wirtschaftsinformatik (vormals Institute of Computer Sciences, man beachte den Plural, eine typisch Bonin'sche Eingebung!) neu nachdenken müssen. Die fachliche Lücke, die dadurch entsteht, dass Deine Stelle auf Wunsch der Hochschulleitung nicht wieder besetzt wird, lässt sich nicht schließen.

Und vor allem Deine engagierte und erfrischende Art wird uns fehlen.

Abschließend wünsche ich Dir, lieber Hinrich, viele weitere glückliche und erfüllte Jahre in Deinem neuen Lebensabschnitt.

Ulrich Hoffmann
Dekan der Fakultät III (Umwelt und Technik)
Leuphana Universität Lüneburg

U. Hoffmann

Lieber Herr Professor Bonin,

es ist mir eine besondere Ehre und Freude, mich als Vorsitzender des Fördervereins Netzwerk Wirtschaft an diesem Wendepunkt Ihres Lebens an Sie wenden zu dürfen.
Für Studierende ist es ein Glücksfall, wenn sie von Lehrenden unterrichtet werden, die Neuem gegenüber stets aufgeschlossen sind, die durch intensive wissenschaftliche Auseinandersetzung mit aktuellen Problemen, durch Polarisierung und kritische Konstruktivität selbst ein Beispiel für lebenslanges Lernen geben.
Ich habe Sie kennen und schätzen lernen dürfen als einen solchen akademischen Lehrer, der direkt heraus und zielstrebig nach der wissenschaftlichen Herausforderung suchend durch seine Denkanstöße und seine begeisternde Vorgehensweise die Studierenden oft zu völlig neuen Projektideen und Erkenntnissen zu inspirieren vermochte.
Sie waren längst schon DORT, wenn andere gerade HIER waren, und es gelang Ihnen immer wieder, Ihre Studierenden durch Ihre unorthodoxen Denkwege mitzureißen und zu motivieren.
Ich höre Sie noch deutlich in einer Ihrer Vorlesungen schmunzelnd sagen: "Standards, Standards, Standards... Sie können sicher sein, meine Damen und Herren, wenn es auf der Welt wieder mal einen neuen Standard für irgend etwas gibt, dann werde ich der Erste sein, der gegen diesen Standard verstößt!"
Damit haben Sie sozusagen Ihren eigenen Standard, den „Bonin-Standard“, aufgestellt, der uns dazu führte, das Vorliegende nicht einfach nur anzuwenden, sondern die Materie zu durchdringen, zu hinterfragen, eigene Methoden zu entwickeln und damit nicht zuletzt dem sich schleichend ausbreitenden Mittelmaß entgegen zu wirken.
Dafür möchte ich Ihnen heute danken, danken auch für die gemeinsame Zeit!
Ich wünsche Ihnen, auch im Namen des Vorstandes und der Mitglieder unseres Vereins von Herzen alles Gute für den nun vor Ihnen liegenden Lebensabschnitt!

Ihr
Stefan Manzke
1. Vorsitzender des
Fördervereins Netzwerk Wirtschaft der Leuphana Universität Lüneburg e.V.

Entwicklung einer Fallstudie für die Lehre im IT-gestützten Personalmanagement

Burkhardt Funk, Mark Lehmann, Peter Niemeyer

ERP-Systeme unterstützen heute eine Vielzahl von Prozessen und Funktionen in Unternehmen. Vor diesem Hintergrund lernen Studierende der Wirtschaftsinformatik an vielen Universitäten während ihres Studiums Konzepte, Technologien und Anwendungsszenarien von ERP-Systemen kennen. Dafür werden häufig SAP-basierte Fallstudien eingesetzt, die Funktionsbereiche wie Materialwirtschaft, Logistik und Controlling abdecken.

Derzeit gibt es keine veröffentlichte auf personalwirtschaftliche Themen fokussierte Fallstudie, die in SAP-Systemen bearbeitet werden kann. Diese Lücke schließt die im Rahmen dieser Arbeit entwickelte Fallstudie. Darüber hinaus wurde ein in der SAP-Lehre neues Fallstudienkonzept entwickelt und eingesetzt.

1. Vorbemerkungen

ERP-Systeme werden heute in allen Tätigkeitsbereichen eines Unternehmens eingesetzt. Deshalb besteht die Notwendigkeit zukünftige Arbeitnehmer bereits während Ausbildung und Studium im Umgang mit solchen Systemen zu schulen. Aus diesem Grund werden seit vielen Jahren SAP-Systeme an Universitäten zu Lehrzwecken eingesetzt. Zur Unterstützung des Einsatzes hat die SAP AG das SAP University Alliance (UA) Programm ins Leben gerufen. Hauptziel des Programms ist die Unterstützung der Ausbildung durch die Bereitstellung neuster SAP-Technologien [9].

In Deutschland die University Competence Center in Magdeburg und München Bestandteil des UA-Programmes. Sie übernehmen den Betrieb und die Wartung von SAP-Systemen für Hochschulen und bieten ihnen weitere Service-Leistungen wie z.B. Dozentenschulungen und Vorlesungsmaterialien an. Zu den Service-Dienstleistungen gehört ebenfalls die Bereitstellung von Fallstudien, die im Rahmen von anwendungsorientierten Lehrveranstaltungen in SAP-Systemen verwendet werden können.

Mit der Bedeutung IT-gestützter Personalwirtschaftssysteme in der Praxis ist auch der Ausbildungsbedarf in personalwirtschaftlich ausgerichteten Studienangeboten gestiegen [1]. Da eine im Wesentlichen auf personalwirtschaftliche Fragestellungen ausgerichtete Fallstudie zur Bearbeitung in SAP ERP derzeit nicht existiert, wurde ein entsprechendes Gemeinschaftsprojekt des UCC Magdeburg mit dem Institut für elektronische Geschäftsprozesse (IEG) der Leuphana Universität Lüneburg mit dem Ziel initiiert, eine solche Fallstudie zu entwickeln.

Ergebnis des Projekts ist eine Fallstudie im Modul SAP ERP HCM auf Basis eines neu entwickelten Fallstudienkonzepts. Dabei wurden konzeptionelle Aspekte bereits etablierter Fallstudien sowie neue Aspekte basierend auf einer zuvor festgelegten Fallstudiendidaktik eingesetzt, um den Bedürfnissen einer im Rahmen der universitären Lehre eingesetzten Fallstudie gerecht zu werden.

Im zweiten Kapitel dieses Beitrages werden grundlegende didaktische Methoden herausgearbeitet und beschrieben. Auf Basis dieser Methoden wird im dritten Kapitel das neu entwickelte Fallstudienkonzept vorgestellt. Abgeschlossen wird der Beitrag mit einer kurzen Zusammenfassung der Ergebnisse, sowie einem knappen Ausblick auf den weiteren Projektverlauf.

2. Grundlagen der Fallstudiendidaktik

Das vorliegende Kapitel beschäftigt sich mit in der Literatur beschriebenen Methoden zur Entwicklung und Konzeption von Fallstudien und deren Einsatz in der Lehre.

2.1 Ziele

Die Verknüpfung von theoretischem Wissen mit praktischen Aktivitäten ist das primäre Ziel der Fallstudienarbeit. Dies kann nur gelingen, wenn die fachlichen Inhalte für die Studierenden von hoher Relevanz sind [4].

Verbunden mit der Fallstudienarbeit ist die Entwicklung der Handlungskompetenz der Fallstudienbearbeiter. Studierende müssen möglichst früh lernen, selbstständig Entscheidungen zu treffen [6]. Nach [2] setzt sich Handlungskompetenz aus der Wissenskomponente, der heuristischen Komponente und der Persönlichkeitskomponente zusammen. Die Wissenskomponente besagt, dass zur Durchführung einer Handlung sowohl vorhandenes Wissen aktiviert als auch benötigte Kenntnisse zur Lösung des Falls angeeignet werden müssen. Heuristiken sind Strategien und Techniken zur Lösung eines Problems und werden in geistige Operationen eingebettet und können erfolgreicher genutzt werden, wenn sie selbstständig erworben werden [10]. Gegenstand der Persönlichkeitskomponente sind persönliche Eigenschaften eines Menschen (z.B. Motivation, Kreativität etc.). Im Rahmen der Fallstudienarbeit sollen verschiedene persönliche Eigenschaften der Fallstudienbearbeiter angesprochen und ausgebildet werden.

2.2 Aufbau

Der Aufbau einer Fallstudie orientiert sich an den drei Unterrichtsprinzipien *Exemplarität*, *Anschaulichkeit* und *Handlungsorientierung*.

Exemplarität ist eng mit dem Ziel verbunden Situationen zu behandeln, die für die Fallstudienbearbeiter von Bedeutung sind. Der behandelte Fall sollte jedoch nicht nur für *jemanden*, sondern auch für *etwas* exemplarisch sein.[8] Dies bedeutet, dass die im Rahmen der Fallstudie behandelte Situation verallgemeinert werden kann.

Unter *Anschaulichkeit* sind gestalterische Aspekte zu verstehen. Verbunden ist dies mit dem Ziel, dass der Fallstudienbearbeiter während der Arbeit angeregt wird, sich mit der behandelten Thematik intensiver auseinander zu setzen. Ein Mittel zur Verwirklichung von Anschaulichkeit stellt die Fallgeschichte einer Fallstudie dar. Dabei wird der Bearbeiter sehr detailliert in die Problematik der Fallstudie eingeführt. So kann er durch die Fallgeschichte unter anderem erfahren, wo die Situation spielt, wer an der Situation beteiligt ist und warum das behandelte Problem überhaupt entstanden ist.

Handlungsorientierung bedeutet, dass die Bearbeiter der Fallstudie sich aktiv am Lernprozess beteiligen, und dass alle Tätigkeiten von der Planung bis zur Kontrolle von derselben Person ausgeführt werden sollten.

2.3 Einsatz

Der Einsatz einer Fallstudie kann in Form eines Entscheidungs- und Problemlösungsprozesses organisiert werden. Als Ausgangsbasis für einen solchen Prozess bietet sich das Ablaufschema nach [5] an. Es setzt sich aus sechs Phasen zusammen und beruht im groben auf der Konfrontation mit dem Fall, der Suche nach Lösungsalternativen, der Begründung der Lösung und dem Vergleich mit einer realen Lösung. Innerhalb der einzelnen Phasen findet Plenumsarbeit und Kleingruppenarbeit statt. Die Phasen Konfrontation, Disputation und Kollation lassen sich idealerweise im Plenum bearbeiten, wohingegen die anderen drei Phasen vor allem bei großen Gruppen in Kleingruppenarbeit behandelt werden sollten.

Das dargestellte Ablaufschema darf nicht als starres Modell verstanden werden. Es stellt lediglich eine Grundstruktur für den Einsatz von Fallstudien dar. Es kann dabei auch Vor- und Rückgriffe auf einzelne Phasen geben, Phasen können mehrmals durchlaufen werden und die Bearbeitung der Phasen unterschiedlich lang sein.

Zur Verdeutlichung der einzelnen Phasen werden ihre Ziele in Tabelle 1 beschrieben.

Tab.1 Ablaufschema für den Einsatz von Fallstudien nach [6]

Phase	**Ziel**
Konfrontation	Erfassung der Probleme, Überblick verschaffen und Ziele formulieren
Information	Vorhandene Informationen bewerten und ggf. weitere Informationen beschaffen
Exploration	Lösungsalternativen suchen
Resolution	Vergleich und Bewertung der Lösungsalternativen
Disputation	Verteidigung der gewählten Lösung
Kollation	Vergleich der eigenen Lösung mit der „realen“ Lösung

2.4 Varianten

Nach [7] haben sich vier Grundvarianten von Fallstudien etabliert. Sie unterscheiden sich in der Darstellung des behandelten Falls, der Informationsbeschaffung bzw. –Bereitstellung und der Problemfindung bzw. –lösung. Tabelle 2 stellt die vier methodischen Varianten und ihre Charakteristika dar.

Anhand der Beschreibung der verschiedenen Fallstudienvarianten und deren unterschiedlicher Schwerpunkte wird deutlich, dass das in Kapitel 2.3 beschriebene Ablaufschema je nach Variante angepasst werden muss. So liegt beispielsweise der Fokus der Stated-Problem-Method bei der Lösungskritik, sodass die letzten beiden Phasen Disputation und Kollation besonders intensiv bearbeitet werden sollten. Wohingegen der Schwerpunkt der Case-Study-Method bei der Problemerkennung liegt und die Phase Konfrontation entsprechend intensiv sein sollte.

Tab.2 Gegenüberstellung der verschiedenen Fallstudienvarianten nach [5]

	Problem-erkennung	**Informations-gewinnung**	**Ermittlung von Lösungs-alternativen und Entscheidung**	**Lösungskritik**
Case-Study-Method	*Schwerpunkt:* Verborgene Probleme müssen analysiert werden	Informationen werden gegeben	Anhand der gegebenen Informationen werden Lösungsalternativen entwickelt und eine Entscheidung getroffen	Vergleich der eigenen Lösung mit der „realen“ Lösung
Case-Problem-Method	Probleme sind ausdrücklich genannt	Informationen werden gegeben	*Schwerpunkt:* Anhand der vorgegebenen Probleme und Informationen werden Lösungsalternativen entwickelt und eine Entscheidung getroffen	Evtl. Vergleich der eigenen Lösung mit der „realen“ Lösung
Case-Incident-Method	Lückenhafte Darstellung des Falls	*Schwerpunkt:* Informationen müssen selbstständig beschafft werden	Auf Basis der gesammelten Informationen werden Lösungsalternativen entwickelt und eine Entscheidung getroffen	Evtl. Vergleich der eigenen Lösung mit der „realen“ Lösung
Stated-Problem-Method	Probleme sind vorgegeben	Informationen werden gegeben	Fertige Lösungen mit Begründungen werden gegeben. Evtl. werden weitere Lösungsalternativen gesucht	*Schwerpunkt:* Kritik der vorgegebenen Lösungen

3. HCM-Fallstudie

3.1 Besonderheiten bei SAP-Fallstudien

SAP-Fallstudien haben insofern Parallelen zu den in Kapitel 2.4 beschriebenen Varianten Case-Problem-Method und Stated-Problem-Method, da in der Regel das Problem klar definiert ist. Dies wird durch den Umstand begründet, dass der Fokus von SAP-Fallstudien nicht bei der Analyse des gegebenen Problems liegt, sondern bei der verwendeten Lösung in SAP.

Im Kern unterscheiden sich SAP-Fallstudien daher von den beschriebenen Fallstudien in den in Kapitel 2.3 beschriebenen Phasen Exploration und Resolution. Bei den in Kapitel 2 beschriebenen Fallstudien handelt es sich stets um mehrere Lösungsalternativen die entweder selbstständig entwickelt oder bewertet werden müssen. Bei den derzeitigen SAP-Fallstudien wird meistens nur eine Lösung betrachtet, wobei diese in der Regel durch den Fallstudienautor vorgegeben wird. Dies ist notwendig, da eine Interaktion mit der Software nur auf bestimmten festgelegten Wegen stattfinden kann, was von den Studierenden häufig als reines „Nachklicken“ empfunden wird.

Zusammenfassend lässt sich basierend auf den Parallelen von SAP-Fallstudien mit den beschriebenen Fallstudienvarianten die in Kapitel 2 beschriebene Fallstudiendidaktik auch auf SAP-Fallstudien anwenden. Auf Basis der Unterschiede muss jedoch eine Anpassung des Ablaufschemas stattfinden.

3.2 Anforderungen

Die Anforderungen an die HCM-Fallstudie lassen sich in fachliche und didaktische Anforderungen unterscheiden.

Aus fachlicher Sicht soll die HCM-Fallstudie dem Bearbeiter einen detaillierten Einblick in die personalwirtschaftliche Arbeit mit SAP ERP geben. Dafür soll der Bearbeiter in das Modul SAP ERP HCM eingeführt werden. Am Ende der Fallstudie soll der Bearbeiter im Umgang mit SAP ERP HCM geübt sein und wichtige Funktionen des Moduls beschreiben und erklären können. Außerdem soll der Bearbeiter in der Lage sein, Zusammenhänge zwischen den verschiedenen Funktionen des Moduls erläutern zu können.

Die HCM-Fallstudie ist für den Einsatz an Universitäten bestimmt und muss daher in ihrem Umfang an die jeweiligen universitären Rahmenbedingungen anpassbar sein. Sie soll in unterschiedlichen Lehrveranstaltungen mit unterschiedlichen Lehrschwerpunkten eingesetzt werden können. Die HCM-Fallstudie soll die Bearbeiter weder über- noch unterfordern. Sie soll lebendig gestaltet werden und zur eigenständigen Auseinandersetzung mit der behandelten Thematik anregen.

3.3 Fallstudiendesign

Im Rahmen des Fallstudiendesigns werden konzeptionelle Entscheidungen zur behandelten Fachthematik und zur Umsetzung der zuvor beschriebenen Fallstudiendidaktik getroffen.

Um den Einsatz der HCM-Fallstudie innerhalb der universitären Lehre zu ermöglichen, darf sie einen gewissen zeitlichen Rahmen nicht übersteigen. Gleichzeitig soll die HCM-Fallstudie möglichst flexibel gestaltet werden. Dadurch kann sie in unterschiedlichen Lehrveranstaltungen mit unterschiedlichen Themen eingesetzt werden. Dies wird dadurch erreicht, dass die HCM-Fallstudie aus mehreren Kapiteln besteht die zum einen möglichst frei miteinander kombiniert werden können und zum anderen eine Bearbeitungszeit von max. 90 Minuten beanspruchen.

Eine Fallgeschichte dient in der HCM-Fallstudie zur Veranschaulichung von Sachverhalten. Innerhalb der verschiedenen Kapitel der HCM-Fallstudie treten in der Fallgeschichte stets dieselben Personen auf, sodass der Eindruck einer durchgängigen Handlung entsteht. Trotzdem sind die Situationen der Fallgeschichte innerhalb der Kapitel abgeschlossen, um den gewünschten flexiblen Einsatz der HCM-Fallstudie zu gewährleisten. Innerhalb eines jeden Kapitels wird ein Themenbereich aus der Personalwirtschaft behandelt, so dass insgesamt die Hauptaufgaben der Personalwirtschaft [3] thematisiert werden.. Da in der Fallgeschichte stets eine Hauptaufgabe der Personalwirtschaft am Beispiel einer Person und eines Unternehmens beschrieben wird, können die Situationen ohne weiteres verallgemeinert werden, sodass dem Prinzip der Exemplarität entsprochen wird.

Eine gute Fallstudie muss die in Kapitel 2.1 beschriebenen Ziele einer Fallstudie erfüllen. Dabei wird für die HCM-Fallstudie davon ausgegangen, dass die Bearbeiter der Fallstudie bereits theoretische Kenntnisse im Bereich der Personalwirtschaft besitzen. Die Ausbildung der Handlungskompetenz fokussiert die Ausbildung von Heuristiken, indem bereits vorhandenes Wissen mit Arbeitsschritten innerhalb von SAP ERP HCM verknüpft wird. Um die Ausbildung von Heuristiken zu fördern, findet vor der eigentlichen Übung eine Wiederholung des vorausgesetzten Wissens statt. Trotzdem stellen eingebettete Übungen einen Großteil der HCM-Fallstudie dar, um die Handlungsorientierung zu gewährleisten. Das Nebenziel der Wissensvermittlung konzentriert sich bei der HCM-Fallstudie auf die Vermittlung von SAP-Kenntnissen. Dafür werden vor einer Übung grundlegende Konzepte und Begriffe erklärt, die im Laufe der Übung angewendet werden. Die durch die Bearbeitung der HCM-Fallstudie erworbenen SAP-Kenntnisse stellen gleichzeitig eine Qualifizierung der Bearbeiter dar.

3.4 Beschreibung

Die Fallgeschichte der HCM-Fallstudie spielt in dem international tätigen IDES-Konzern (Beispielfirma der SAP AG), welcher Produkte fertigt und vertreibt. Im Rahmen der

Fallgeschichte führen die Bearbeiter verschiedene Tätigkeiten in der Personalabteilung des Konzerns durch.

Aus Gründen der Praktikabilität wurde die HCM-Fallstudie in eine Einführungsfallstudie und eine Erweiterungsfallstudie aufgeteilt.

Die HCM-Fallstudie besteht aus insgesamt neun Kapiteln, in denen jeweils ein personalwirtschaftlicher Prozess durchgeführt wird. Die Einführungsfallstudie umfasst die Kapitel

- Organisationsmanagement
- Personaladministration

und stellt die Basis für die Erweiterungsfallstudie dar. Dies bedeutet, dass zunächst beide Kapitel der Einführungsfallstudie bearbeitet werden müssen, damit die Erweiterungsfallstudie bearbeitet werden kann.

Der zweite Teil umfasst die restlichen Kapitel der HCM-Fallstudie. Die Kapitel der Erweiterungsfallstudie können nach Belieben eingesetzt und in ihrer Reihenfolge vertauscht werden:

- Personalbeschaffung
- Personalzeitwirtschaft
- Reisemanagement
- Personalabrechnung
- Personalentwicklung
- Performancemanagement
- Personalcontrolling

Alle neun Kapitel sind nach einer einheitlichen Struktur aufgebaut, um die Übersichtlichkeit der Kapitel zu erhöhen und einen gleichbleibenden Lernprozess zu verwirklichen. Abbildung 1 stellt den Aufbau der Kapitel dar.

1. Personalbeschaffung
1.1 Einleitung
1.2 Vorbereitende thematische Auseinandersetzung
Aufgabenstellungen
Lösungen
1.3 Umsetzung in SAP
1.4 Übung im System
1.5 Zusammenfassung

Abb.1 Kapitelstruktur der HCM-Fallstudie

Am Anfang eines Kapitels wird in der Einleitung eine kurze Beschreibung des im Rahmen des Kapitels behandelten personalwirtschaftlichen Themenbereichs gegeben, die

durch die Einsatzmöglichkeiten eines Personalinformationssystems (PIS) im entsprechenden Themenbereich ergänzt wird.

Der darauffolgende Abschnitt beschäftigt sich mit der Vorbereitung auf die spätere Übung. Dabei handelt es sich um Fragen, die zum einen zur Wiederholung dienen und zum anderen den Bearbeiter auffordern sich Gedanken zu machen, wie die Thematik konkret in einem PIS abgebildet werden kann. Zu den gestellten Fragen werden Lösungsvorschläge gegeben die für einen Vergleich der erarbeiteten Lösungen verwendet werden können.

Im dritten Abschnitt eines Kapitels wird die Umsetzung der Thematik in SAP beschrieben. Zu Beginn wird die Integration der Komponente innerhalb von SAP ERP HCM grafisch dargestellt. Aus Gründen der Übersichtlichkeit wird nur der Zusammenhang der relevanten Komponenten des Kapitels präsentiert. Den Bearbeitern soll dadurch verständlich gemacht werden, wie die verschiedenen Komponenten des Moduls SAP ERP HCM interagieren. Anschließend werden wichtige Konzepte und Begrifflichkeiten erklärt.

Der vierte Abschnitt stellt die eigentliche Übung in SAP dar. Zu Beginn wird die Situation der Fallgeschichte beschrieben, auf die der Bearbeiter in dem Kapitel trifft. Die Fallgeschichte umfasst: Anweisungen, welche Tätigkeiten der Bearbeiter im Folgenden ausführen muss, die beteiligten Akteure der Situation und die Rolle, in der der Bearbeiter in dieser Situation agiert.

Den Abschluss eines Kapitels stellt eine knappe Zusammenfassung dar. Darin werden die im Rahmen der Übung getätigten Handlungen resümiert.

Die verschiedenen Kapitel der HCM-Fallstudie können entweder in Gruppen- oder Einzelarbeit bearbeitet werden. Idealerweise sollte die Übung jedoch in Einzelarbeit durchgeführt werden, um den Lerneffekt der Einzelnen durch die eigene Arbeit im System zu steigern.

Das in Kapitel 2.3 beschriebene Ablaufschema muss aufgrund des Aufbaus der HCM-Fallstudie angepasst werden. Abbildung 2 stellt einen beispielhaften Ablauf für die Arbeit mit der HCM-Fallstudie dar, wobei dieser für jedes Kapitel wiederholt wird.

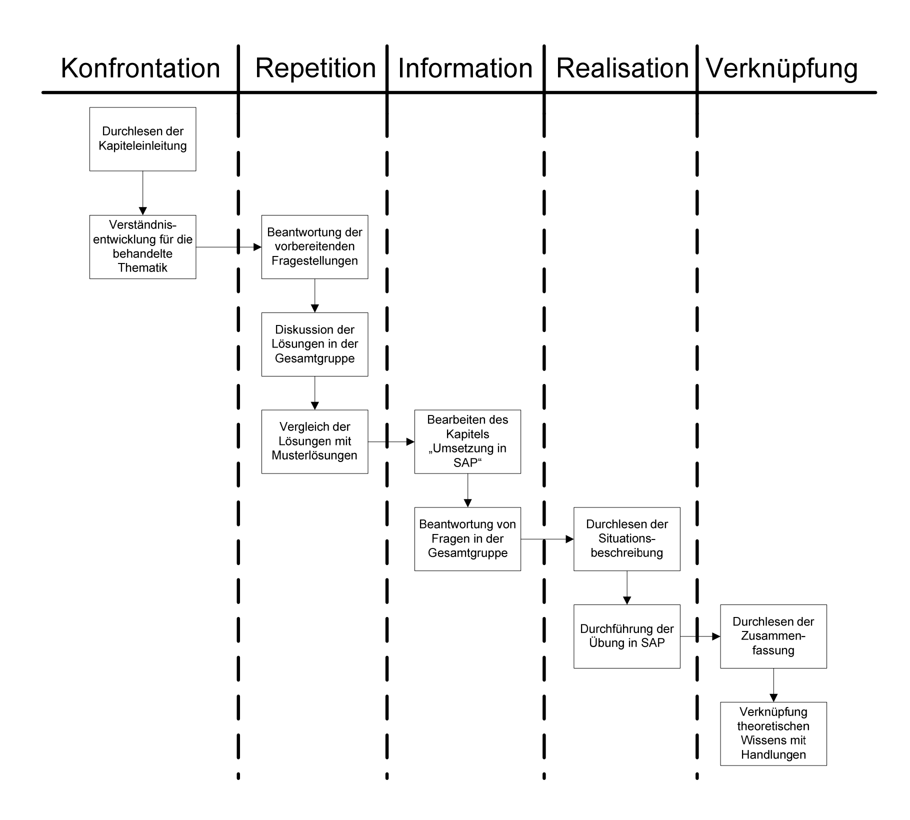


Abb.2 Beispielhaftes Ablaufschema zum Einsatz der HCM-Fallstudie in Anlehnung an [7]

3.5 Ausgewähltes Kapitel

Zur Veranschaulichung des in Kapitel 3.3 und 3.4 beschriebenen Fallstudienkonzepts wird an dieser Stelle beispielhaft das Kapitel „Organisationsmanagement“ aus der Einführungsfallstudie vorgestellt.

In der Einleitung des Kapitels „Organisationsmanagement“ wird im ersten Teil auf das Bedürfnis eingegangen, die Mitarbeiter eines Unternehmens mit ihren Aufgaben, Befugnissen und Verantwortungen in einer Struktur abzubilden. Der zweite Teil der Einleitung befasst sich mit der Notwendigkeit, eine entsprechende Struktur in einem Personalinformationssystem (PIS) abbilden zu können und die Struktur für Auswertungen im Reporting zu verwenden.

Während der vorbereitenden thematischen Auseinandersetzung wird der Bearbeiter mit drei Fragestellungen zum Thema Organisationsmanagement konfrontiert. Bei der ersten Frage wird der Bearbeiter aufgefordert, eine mindestens vierstufige Unternehmensstruktur zu zeichnen. In der zweiten Frage befasst sich der Bearbeiter mit den Daten, die benötigt werden, um eine Unternehmensstruktur in einem PIS abzubilden. Dabei kann die Lösung der ersten Frage eine Hilfestellung für die Lösung der zweiten Frage darstellen. Die dritte Frage beschäftigt sich mit Besonderheiten, die beachtet werden müssen, wenn ein Unternehmen verschiedene Standorte in unterschiedlichen Ländern hat. Da für die Frage mehrere Lösungen (z.B. unterschiedliche Zeitzonen, verschiedene Währungen, unterschiedliche Feiertage etc.) möglich sind, sollen die Bearbeiter an dieser Stelle eigene Erfahrungen und Ideen einbringen. Um einen Vergleich

der erarbeiteten Lösungen zu ermöglichen, werden innerhalb des Abschnitts Musterlösungen angeboten.

Im dritten Abschnitt des Kapitels „Organisationsmanagement“ werden wichtige Konzepte und Begriffe in SAP erläutert. Es wird beschrieben, wie die organisatorische Zuordnung eines Mitarbeiters über die drei Ebenen Unternehmensstruktur, Personalstruktur und Aufbauorganisation erfolgt. Dafür werden wichtige Begriffe der Unternehmensstruktur wie z.B. Mandant, Buchungskreis etc. erklärt, sowie beschrieben, welche Aussage die Unternehmensstruktur in SAP. Es werden außerdem die in SAP verfügbaren Objekte (z.B. Planstelle, Person etc.) und deren Verbindungen innerhalb der Aufbauorganisation erklärt, mit denen der Bearbeiter im Folgenden vierten Abschnitt arbeiten wird.

Die eigentliche Übung des Kapitels befindet sich im vierten Abschnitt. Zunächst erfährt der Bearbeiter, welche Aufgaben während der Übung auf ihn zukommen. Er erfährt, dass er als Mitarbeiter der Personalabteilung eine neue Organisationseinheit mit zwei Planstellen anlegen soll, wobei die eine Planstelle als Leitung der neuen Organisationseinheit eingerichtet wird. Die gesamte Übung findet in der Transaktion PPOME statt. Zur Veranschaulichung von Tätigkeiten im System werden Screenshots, sowie kursive und fette Schrift verwendet.

In der Zusammenfassung wird resümiert, welche Tätigkeiten der Bearbeiter während des vierten Abschnitts ausgeführt hat. Es wird beschrieben, dass im Konzern IDES AG eine neue Organisationseinheit mit zwei Planstellen basierend auf unterschiedlichen Stellen angelegt wurde.

4. Fazit

Das im Rahmen dieses Beitrags vorgestellte Fallstudienkonzept stellt einen innovativen Ansatz zum Einsatz von SAP ERP HCM in der universitären Lehre dar. Der Aufbau der HCM-Fallstudie in Kapitel, die beliebig kombiniert werden können, ermöglicht eine optimale Anpassung an eine Lehrveranstaltung. Außerdem wird der zeitliche Rahmen der einzelnen Kapitel auf die Bedürfnisse der universitären Lehre zugeschnitten.

Das Konzept löst sich von dem Gedanken, eine SAP-Fallstudie ausschließlich als Arbeit in SAP zu interpretieren. Vielmehr wird durch das Ablaufschema ein gesamter Lernprozess abgebildet. Dabei ist die Wiederholung von erworbenem Wissen genauso eingebunden wie der Neuerwerb von Wissen. Um den aktiven Charakter der Fallstudie zu erhalten, findet ein Großteil der Arbeit in SAP statt, wobei das zuvor wiederholte und neuerworbene Wissen angewandt wird.

Im Rahmen der HCM-Fallstudie können verschiedene Unterrichtsmethoden eingesetzt werden. So können einzelne Teile der HCM-Fallstudie in Gruppen- oder Einzelarbeit bearbeitet werden, andere wie z.B. die vorbereitende thematische Auseinandersetzung können in die Vorlesung, in der die HCM-Fallstudie eingesetzt wird, integriert werden.

Eine Evaluation der HCM-Fallstudie ist für das Wintersemester 2009/2010 mit Masterstudierenden der Personalwirtschaft und einer Expertengruppe geplant.

5. Literaturverzeichnis

[1] Bedell MD, Floyd BD, McGlashan Nicols K, Ellis R (2007) Enterprise Resource Planning Software in the human resource classroom. Journal of Management Education 31(1): 43-63.

[2] Brettschneider V (2000) Entscheidungsprozesse in Gruppen: Theoretische und empirische Grundlagen der Fallstudienarbeit. Klinkhardt, Bad Heilbrunn/Obb.

[3] Jung H (2006) Personalwirtschaft. Oldenbourg, München.

[4] Kosiol E (1957) Die Behandlung praktischer Fälle im betriebswirtschaftlichen Hochschulunterricht (Case Method). Duncker & Humblot, Berlin.

[5] Kaiser FJ (1973) Entscheidungstraining: Die Methoden der Entscheidungsfindung; Fallstudie, Simulation, Planspiel. Klinkhardt, Bad Heil brunn/Obb.

[6] Kaiser FJ (1983) Die Fallstudie: Theorie und Praxis der Fallstudiendidaktik. Klinkhardt, Bad Heilbrunn/Obb.

[7] Kaiser FJ, Kaminski H (1997) Methodik des Ökonomieunterrichts: Grundlagen eines handlungsorientierten Lernkonzepts mit Beispielen, Klinkhardt, Bad Heilbrunn/Obb.

[8] Reetz L, Beiler J, Seyd W (1993) Fallstudien Materialwirtschaft: Ein praxisorientiertes Wirtschaftslehre-Curriculum. Feldhaus, Hamburg.

[9] Rosemann M, Maurizio AA (2005) SAP-related Education – Status Quo and Experiences. Journal of Information Systems Education 16(4): 437-453.

[10] Weitz BO (1996) Fallstudienarbeit in der beruflichen Bildung. Gehlen, Bad Homburg.

Closures, Continuations und die Java Virtual Machine

Karl Goede

1 Einführung

Closures und Continuations gehören zu den eher esoterischen Konzepten, die Programmiersprachen aufweisen können. Es verwundert daher nicht, dass sie zuerst in Programmiersprachen auftraten, die abseits des Mainstreams liegen (Stichwort LISP/Scheme). Tatsächlich interessierten sich eher bestimmte Mathematiker (unter heutigen Verhältnissen: „theoretische Informatiker") und Programmierer mit LISP-Faible für deren Nutzung. Verwunderlich ist es aber, dass auch Programmiersprachen solche Konzepte kennen, die sich vorwiegend an den „flinken Praktiker" wenden. Dazu gehören Perl, Python und Ruby, aber auch Frameworks wie Seaside (ein Smalltalk Framework).

Immer wieder kommt die Diskussion auf, ob Closures und Continuations nicht unbedingt in virtuelle Maschinen aufgenommen werden sollten (s. z. B. [4]). Dies könnte dem „Normalprogrammierer" gleichgültig sein, solange nur virtuelle Maschinen (kurz: VMs) wie Parrot oder YARV betroffen sind oder Microsoft sein .NET unter den Stichworten „Iron Python" , „Iron Ruby" [6] diesbezüglich verstärkt. Nicht mehr gleichgültig ist es, wenn unter diesem Aspekt auch die JVM (Java Virtual Machine) ins Fadenkreuz der Verbesserer gerät.

Im Folgenden wird nach einer kurzen Betrachtung des eventuellen Nutzens von Closures und Continuations die daraus resultierende Belastung für virtuelle Maschinen und Laufzeitbibliotheken betrachtet. Es stellt sich dabei auch heraus, dass die Namen „Closures" und „Continuations" in gängigen Implementierungen nur sehr eingeschränkt halten, was sie versprechen. So eingeschränkt, wären „Continuations" für VMs aber immerhin tragbar. Solch eine Aussage gilt nicht für Closures - selbst wenn man sich auf die Leistungsfähigkeit der Closures etwa von Ruby beschränkt, die gerade ausreicht, anonyme Funktionen mit lokalen Variablen zu realisieren.

Man könnte als Mainstream-Programmierer die genannten Konzepte wie böhmische Dörfer links liegen lassen, sofern sie nicht die Programmiersprache belasten. Tatsächlich gibt es aber eine Belastung. Wie groß die wirklich ist und ob sie den Nutzen lohnt, wird im Folgenden untersucht.

2 Closures

Closures werden für die Bereitstellung anonymer Funktionen gebraucht. Sofern es sich nicht um literal abgeschlossene Funktionen der Art

```
def add(a, b, c) a + b + c  end  # eine Ruby-Methode
```

bzw.

```
var add (a: Int, b: Int, c: Int) => a + b + c  // eine Scala-Funktion
```

handelt, stehen dahinter „einzufrierende" Blockaktivierungen. Sie müssen über spätere Veränderungen im fortlaufenden Programm hinweg aufbewahrt werden. Eigentlich gilt das auch für den Kontext (insbesondere den Inhalt des Heap), in dem eine Funktionsdefinition stattfand. Lästige Komplikationen dieser Art werden aber gern ignoriert oder wegdiskutiert. Implementiert wird so, wie es in den Compiler bzw. Interpreter ohne Zusatzaufwand einbaubar ist.

Typisch (in seiner Sinnlosigkeit) für den einleitend genannten esoterischen Charakter von Closures ist ein (das einzige!) Beispiel im „Standardhandbuch" Pickaxe [15] für Ruby :

```
def nTimes(aThing)
```

```
  return proc[1] { |n| aThing * n }
end

p1 = nTimes(23)
p p1.call(3)     # » 69
p p1.call(4)     # » 92
p2 = nTimes("Hello ")
p p2.call(3)     # » "Hello Hello Hello "
```

Im Kommentar (durch # eingeleitet) sind die Ausgaben des Programms dargestellt. Beim Ablauf (in einer Datei namens *Closures.rb* festgehalten) zeigt der Bildschirm Folgendes:

```
>ruby Closures.rb
69
92
"Hello Hello Hello "
>Exit code: 0
```

Ein Hauch von Sinn läßt sich mit viel Mühe im folgenden - allerdings auch nicht bahnbrechendem - Beispiel erkennen (die Ausgaben sind wiederum als Kommentar angefügt)[2]:

```
# Closure-Beispiel /Goe.

def temperaturbasis(grad, hinweis)
  b = " Grad " + hinweis
  proc { |n| [grad + n, b] }
end

kelvin = temperaturbasis(273, "Kelvin")
print kelvin.call(20), "\r"    # » 293 Grad Kelvin
print kelvin.call(-42), "\r"     # » 231 Grad Kelvin
celsius = temperaturbasis(0, "Celsius")
print celsius.call(20), "\r"     # » 20 Grad Celsius
print kelvin.call(-272), "\r"    # » 1 Grad Kelvin
```

Die Variablen *kelvin* und *celsius* besitzen dabei als Werte Funktionsobjekte. Mit der Methode *call* läßt sich deren Funktion ausführen. Im Beispiel werden die Funktionsobjekte von der Methode *temperaturbasis* erzeugt und als Wert zurückgeliefert. Die aktuellen Parameterwerte 273, " Kelvin" bzw. 0, " Grad Celsius" werden für die spätere Verwendung „eingefroren".

Eine kleine Modifikation des Beispiels (*p* anstelle von *print* nutzt die komfortablere Ausgabetechnik des in Ruby eingebauten „Reflection Interface") zeigt, dass Ruby die Umgebung der Methode *temperaturbasis* nicht mit einfriert:

```
# Closure-Beispiel mit globaler Variabler (das Attribut @hinweis)

@hinweis = "undefiniert"
def temperaturbasis(grad)
  return proc { |n| [grad + n, @hinweis] }
end
@hinweis = "Kelvin"
kelvin = temperaturbasis(273)
p kelvin.call(20)    # » [293, "Kelvin"]
p kelvin.call(-42)     # » [231, "Kelvin"]
@hinweis = "Celsius"
celsius = temperaturbasis(0)
```

[1] Anstelle von „proc" darf auch die Bezeichnung „lambda" verwendet werden, um Liebhabern des λ-Kalküls einen gewissen Wiedererkennungswert zu vermitteln.

[2] Mit etwas „methodischem Wahnsinn" ließe sich das Beispiel für eine Welt mit n Temperaturskalen auf Basis verschiedener Nulllagen ausbauen, wenn diese etwa aus einem Internet-Wiki ausgelesen würden und *temperaturbasis* daraus n angepasste anonyme Funktionsobjekte erzeugte und Elementen eines Arrays zuwiese.

```
p celsius.call(20)      # » [20, "Celsius"]
p kelvin.call(-272)     # » [1, "Celsius"]
```

So führt der Aufruf *p kelvin.call(-272)* zur falschen Beschriftung *"Celsius"* . Allerdings würde das Aufbewahren von Umgebungsinformationen zu einer unerträglichen Belastung der Laufzeit führen[3]. Insofern verhält sich Ruby äußerst vernünftig. Für Programmierer ergibt sich eine einfache Regel: Wenn tatsächlich in der Umgebung enthaltene Werte aufbewahrt werden sollen, müssen sie explizit lokalen Variablen zugewiesen werden. Es resultiert eine Fehlerquelle, wenn die Regel ignoriert wird.

Das genannte „Closure"-Verhalten bezüglich globaler Variabler ist keineswegs ein nur Ruby eignendes „Dünnbrettbohrverhalten". Beispielsweise regelt es die Programmiersprache Scala nicht anders. Erfreulicherweise wird dieser Sachverhalt für Scala jedoch sauber dokumentiert [10][4].

Was den Nutzen von Closures betrifft, so gibt es neben reihenweise künstlichen Anwendungsbeispielen auch vernünftige Effekte. So lassen sich Codeverdoppelungen (z. B. in Case-Listen, bei denen Fälle stückweise gleich zu behandeln sind) vermeiden. Allerdings ist es dem Verfasser nicht gelungen, ad hoc Beispiele zu finden, die dabei nicht besser, eleganter, mit weniger Nachdenken - und erheblich effizienter - durch den Einsatz lokaler Funktionen darstellbar sind.

Im dem folgenden aus [5] entnommenen Beispiel würden Codeverdoppelungen auftreten, wenn innerhalb *main* ein switch-Konstrukt eingefügt würde, bei dem in manchen der „cases" eine Ausgabe des zu *main* lokalen Arrays a stattfinden soll:

```
import std.stdio;
void main ( ) {
  int[9] a;   // standardmässig mit Nullen initilialisiert

  void a_detailliert() { // lokale Funktion a la Pascal
  writefln("Array a:");
  foreach (i, element; a) writefln("a[%d] = %4d", i, element);
  writefln("-------------------------------");
  }
  for (int i = 0; i < a.length; i++) a[i] = 10 * i;
  // falls man auf foreach verzichten möchte
  a_detailliert();
  int[] s = a[4..7];
  writefln(a, s);
  s[1] = 4711;
  writefln(a, s); // ... auch a[5] == 4711

  int zähler = 0; // deutsche Umlaute zulässig - dank UTF-8
  foreach (ref element; a) {element = zähler; zähler += 100; }
  writefln(a);
}
```

Statt jedesmal explizit den Code für die Ausgabe von *a* aufzuführen, wäre dann entsprechend oft der Aufruf *a_detailliert()* einzusetzen. Mit einer außerhalb *main* liegenden Funktion (bzw. Methode im Rahmen einer Klasse) gäbe es den Zugriff auf das lokale Array *a* nicht. Dieses Array müsste dann global zur betrachteten Funktion (hier also *main*) definiert bzw. im Rahmen einer Klasse als Attribut organisiert werden.

Wesentlich ist hier die in Abb. 1 dargestellte Stack-Technik, bei der eine lokale Funktion *lf* die Blockaktivierung der sie enthaltenden Funktion *f* mit benutzen kann.

Dabei sind für das Laufzeitsystem keinerlei besondere Vorkehrungen nötig. Es genügen die ohnehin für potentiell rekursive Funktionen üblichen Standards der Blockaktivierungen mit effizienter Freigabetechnik unter simplem Zurücksetzen des Stackzeigers.

---

[3] Dies wird noch näher bei der Untersuchung der Continuations (s. u.) diskutiert.

[4] Dort wird mit viel Sinn für Marketing formuliert. So heißt es auf S. 152: „Intuitively, Scala's closures capture variables themselves, not the value to which variables refer."

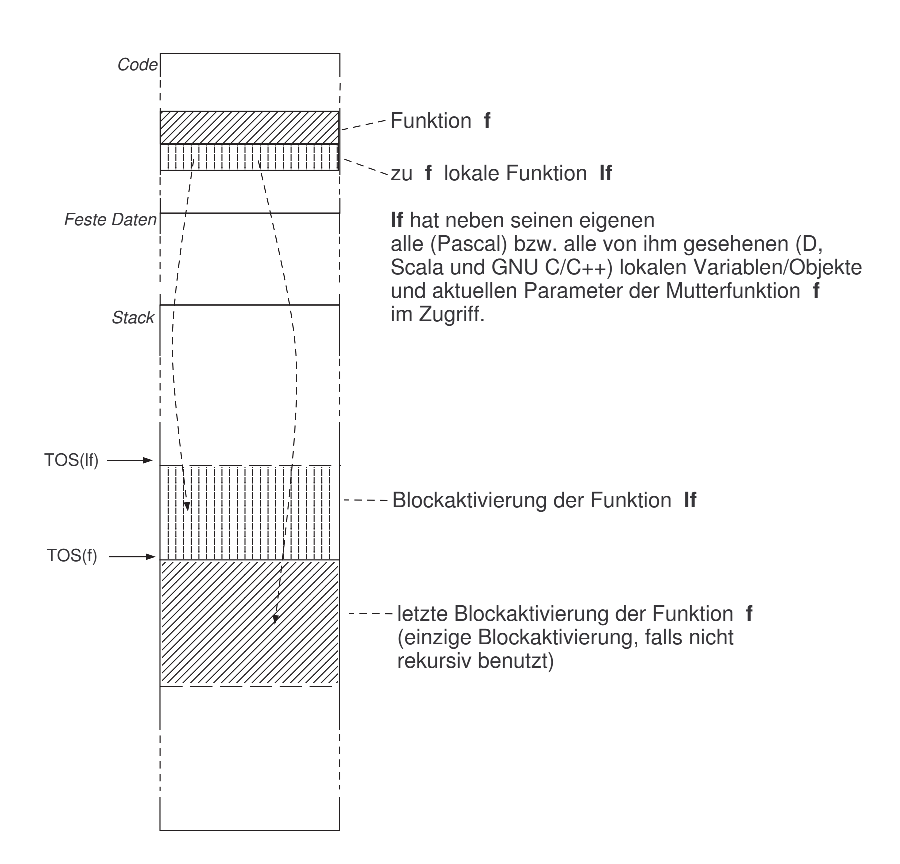


Abbildung 1: Speichersituation nach Aufruf der lokalen Funktion lf

3 Continuations

Eine *Continuation* bezeichnet für ein Programm P zu einem bestimmten Zeitpunkt t die vollständige Information darüber, wie P fortzusetzen und - letztlich - zu Ende zu bringen ist. Bei Programmiersprachen wie Pascal oder C enthält die Continuation beispielsweise konkret Folgendes:

- Adresse des nächsten auszuführenden Befehls
- vollständiger momentaner Speicherzustand, d. h.
  - fester Speicher, darunter die Werte der auf Programmebene definierten Variablen, sofern diese Variablen nicht vom Compiler als „Grundlast" auf dem Stack ab BOS (Bottom Of Stack) gelegt werden
  - Stack und momentaner Stackzeiger (TOS = Top of Stack)
- die gerade benutzten Teile des Heap

Tatsächlich weisen praktisch alle gängigen Programmiersprachen eine bestimmte Art von Continuation-Konzept auf. Continuations definieren dabei mögliche Programmfortsetzungen nach Eintritt einer Exception. Sie müssen dazu als Rückzugspositionen explizit definiert werden. Allerdings wird an der Rückzugsposition üblicherweise nur jener Teil des Speicherzustands gesichert, der „billig zu haben ist" - nämlich der gerade gültige Stackzeiger TOS(r).

Die für Continuations relevanten Probleme lassen sich besonders einfach am Speicherabbild eines gerade ablaufenden C-Programms beleuchten[5], da dabei Continuations-Effekte in Multithreading-

[5]Die Programmiersprache Scala kennt seit Vs. 2.8 ein Analogon dazu (s. [11]). Den C-Konstrukten *setjmp* und *longjmp* entsprechen - natürlich mit „syntactic sugar" versehen - die Scala-Funktionen *shift* und *reset*. Die von Scala benutzte JVM „merkt" davon allerdings nichts. Beide Scala-Funktionen werden per Compiler-Library abgedeckt. Insofern würde die Betrachtung von Scala keine zusätzlichen Erkenntnisse bringen.

Situationen nicht betrachtet zu werden brauchen[6]. Es gibt im Detail reichlich Unterschiede im Vergleich zur Speicherorganisation der JVM. So werden z. B. in C-Implementierungen üblicherweise Control Stack und Datenstack nicht separiert, sondern in einem einzigen Kombinationsstack zusammengefasst. Sehr wohl werden aber die Kernprobleme sichtbar, die bei einem Rückkehrwunsch von einem jüngeren in einen älteren Programmzustand auftreten. Insbesondere ist zu sehen, wie billig ein „Rückfall" auf dem Stack zu haben und wie teuer die Angelegenheit ist, wenn ein älterer Speicherzustand vollständig wieder hergestellt werden muss, um „klinisch sauber" dort das Programm fortsetzen zu können. Mit anderen Worten: Es ist sehr einfach zu sehen, was passieren muss, wenn analog zum Ablauf in einem Datenbanksystem nach einer abgebrochenen Transaktion so fortgesetzt werden muss, als wäre zwischenzeitlich „nichts geschehen".

Im C-Fall geschieht die schnelle Rückkehr in einen älteren Zustand m. H. eines „Longjump". Dazu wird zum Zeitpunkt *r* die Adresse PC(r) des nächsten auszuführenden Befehls nach dem „Rückfall" und der aktuelle Stackpointer TOS(r) an geeigneter Position[7] m. H. der *setjmp*-Anweisung festgehalten. Das Programm kann m. H. der longjump-Anweisung später zu einem Zeitpunkt *a* (für *a*ktueller Zeitpunkt) bei Bedarf an der Stelle *r* fortgesetzt werden. Die normale Rückabwicklung des Stack wird dabei außer Kraft gesetzt. Stattdessen wird schlagartig der gesamte Keller von TOS(r) bis TOS(a) (s. Abb. 2) freigegeben, indem einfach der Wert des Zeiger TOS(a) durch jenen von TOS(r) ersetzt wird[8].

Prototypisch sieht das Ganze folgendermaßen aus:

```
static jmp_buf err_env;  // Zustandsinfo für Fehlerwiederanlauf
  ...
  ...
r:  if (i = setjmp(err_env)) // Fehlerfänger setzen
switch(i)
{
case 0:  break;
case 7:  ...; /* Fall „7" behandeln */ break;
default: ...; /* sonstiges Handling */ break;
}
  ...
   ...
   // tief in der Aufrufhierarchie
   ...
 a:  if (err_env) longjmp(err_env, 7);
   ...
```

Die globale Variable *err_env* vom Typ *jmp_buf* (in einer compilerspezifischen Header-Datei definiert) nimmt dabei die Rücksprungadresse PC(r) und den Rückzugskellerzeiger TOS(r) in einer für den Prozessor und das Laufzeitsystem geeigneten Art und Weise auf. Die Anweisung *longjmp(err_env, 7);* führt unter Umgehung der Standard-Rückabwicklung zurück zur mit *r:* markierten Zeile. *setjmp(err_env)* liefert nicht wie beim Setzen der „Rücksprungfalle" den Wert 0, sondern den zweiten Parameter von *longjmp* (also 7).

Wollte man den Programmierern eines Datenbanksystems die Realisierung des Transaktionsmanagements besonders leicht machen, ließe sich die Continuation-Problematik aus Sicht einer heutigen Programmiersprache noch gewaltig steigern. Es wäre der in Abb. 2 dargestellte Speicherzustand um den auf das Programm bezogenen Zustand externer Speicher zu erweitern. Die Krönung bestünde darin, neben aktuell zugänglichen Festplatten noch Sicherungsplatten bzw. Sicherungs-Magnetbänder mit ins Kalkül zu ziehen.

Im Falle von Sprachimplementierungen mit automatischer garbage collection (also fast alle bis auf u. a. C/C++) würde indirekt mit dem Rücksetzen des Kellers auch den Heap um seit *r* neu angelegte

[6] Interessant wäre eine Untersuchung, ob es irgendwo eine korrekte Implementierung von Continuations in einem Multithreading-Umfeld in irgendeiner Programmiersprache gibt. Koichi Sasada, der Entwickler von YARV (Yet another Ruby VM; ein bytecode interpreter) verneint die Korrektheit für Ruby [13]. Ruby 1.9, für das YARV einen massiven Performancegewinn gegenüber Ruby 1.8 bringt, hatte dementsprechend Continuations abgeschafft. Gegen seinen hinhaltenden Widerstand wurden Continuations in Vs. 1.9 später wieder „hineingezwungen".

[7] Normalerweise befindet sie sich im Bereich der Schnittstelle zum Betriebssystem, um auch nach einem Interrupt als mögliche Fortsetzungsadresse nutzbar zu sein.

[8] Je nach Prozessortechnologie und Laufzeitsystem (durch Libraries gegeben) ist die Abwicklung geringfügig komplizierter. In jedem Fall werden nur Registerwerte umjustiert und der Programmcounter auf den für die Rückkehr nach *r* festgehaltenen Wert gesetzt.

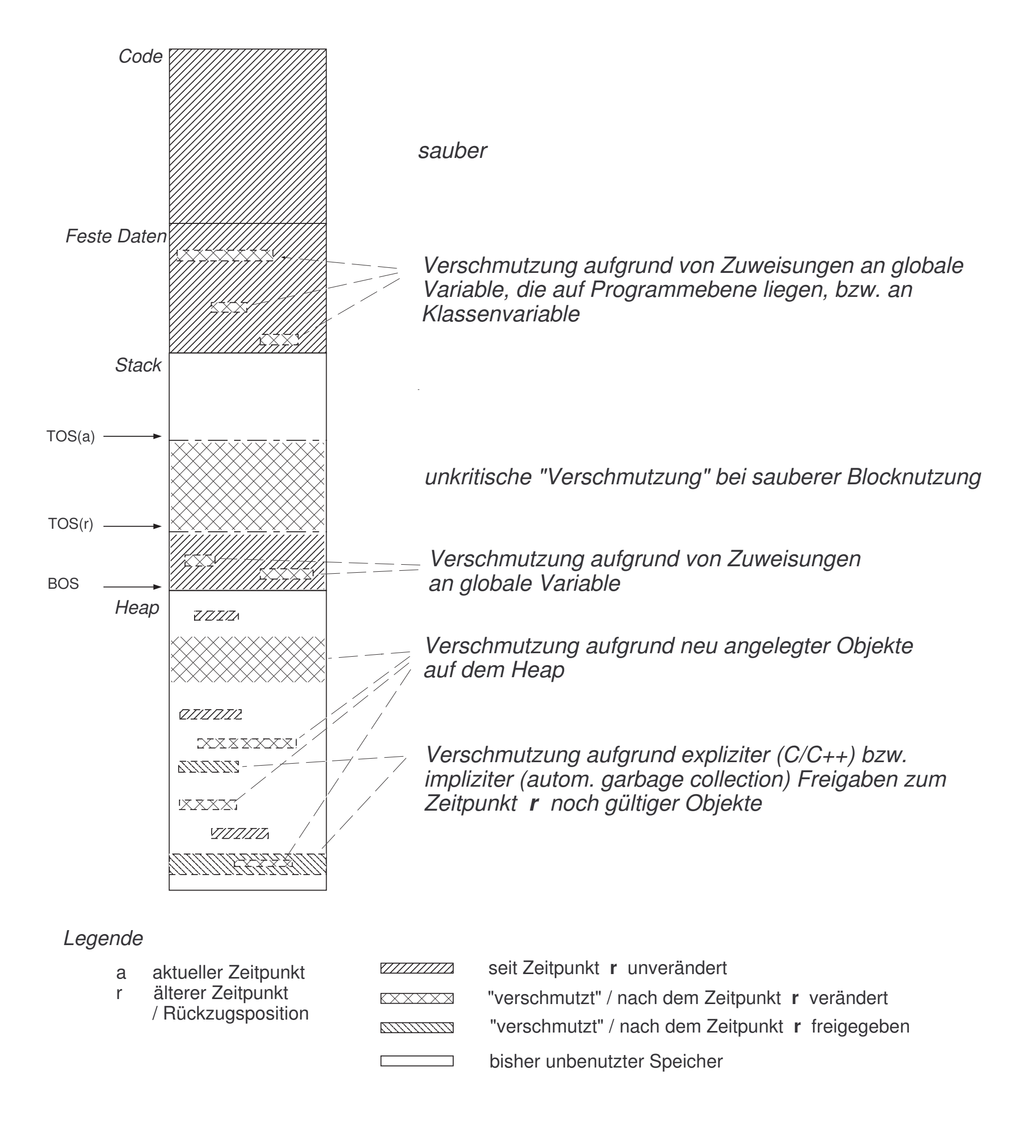


Abbildung 2: Programmbezogener Hauptspeicherzustand im Continuation-Fall

Objekte befreien. Sie werden ja automatisch unzugänglich, sofern sie nicht noch mit Referenzzählern arbeiten. Probleme bereiten dann nur noch jene Objekte, die seit r unzugänglich geworden sind und vom garbage collector bereits „weggeräumt" worden sind.

Eine saubere Implementierung von Continuations muß alle in Abb. 2 aufgeführten „Verschmutzungen" des Speicherzustands bezüglich des Zeitpunkts r beseitigen. Die oben dargestellte Zurücknahme des „Stackschmutzes" ist dabei eine Bagatelle im Vergleich zur Beseitigung anderer „Verunreinigungen". Folgendes wäre insgesamt zu tun:

- Stack auf r zurücksetzen
- Änderungen globaler Variablen im Bereich der festen Daten und auf dem Stack anhand einer zum Zeitpunkt r anzufertigenden Sicherungskopie zurücknehmen
- Änderungen auf dem Heap (incl. Freigaben!) anhand einer zum Zeitpunkt r anzufertigenden Sicherungskopie zurücknehmen

Am unangenehmsten ist der letzte Punkt.

Im Falle „handgesteuerter" Freigaben à la C/C++ müssten die zum Zeitpunkt r auf dem Heap liegenden Objekte relozierbar sein oder auf exakt den ursprünglichen Positionen rekonstruiert werden. Möglicherweise werden sie vom Stack bzw. den festen Daten aus über die alten Zeiger angesprochen werden. Sie können auch untereinander verzeigert (ebenfalls mit den alten Adressen) sein. Mit den üblichen weitgehend autonom agierenden Heap-Verwaltungen ist Letzteres nicht zu machen. Es bleibt nur die „Ochsentour" des Relozierens: Alles mit Surrogatadressen sichern (mit Tiefenkopie!) und beim

Rückladen anhand der gerade verfügbaren Heap-Positionen die Verzeigerung neu organisieren. Das hieße:

- Zum Zeitpunkt r muss ein tiefer Marshalling-Prozess ablaufen
- Zum Zeitpunkt a wird ein gegenläufiges tiefes Unmarshalling benötigt

Für das Marshalling/Unmarshalling (alternativ wird für das Marshalling auch gern die Vokabel „Reification" benutzt) gibt es von Perl über Ruby bis Java und schließlich Scala fertige Library-Funktionen. Gleichwohl ist beides aber für Continuations-Zwecke sehr unangenehm zu organisieren und außerdem derart zeitaufwendig, dass Programmiersprachen mit brauchbarem Performanceanspruch dies auf keinen Fall tun. Continuations degenerieren damit - wie auch Closures - im Umfeld der genannten Programmiersprachen zu „Discountkonstrukten".

Im Falle automatischer garbage collection müssten seit r „weggeräumte" Objekte reinstalliert werden. Es ergibt sich im Effekt die gleiche Problematik wie beim oben dargestellten Fall der „Handsteuerung"

Immerhin wäre ein sauberes Continuation-Handling möglich, wenn

- zur Speicherung ausschließlich der Stack (und im C-Umfeld der Bereich fester Daten) benutzt wird,
- globale Daten nicht modifiziert werden.
- die Heap-Nutzung so eingeschränkt wird, dass die oben dargestellten Marshalling/Unmarshalling-Maßnahmen überflüssig werden

Der letzte Punkt, also die Einschränkung der Heap-Nutzung würde die Programmierung beispielsweise in Ruby, Java und Scala stark erschweren [9].

Es gibt also für reale Programmiersprachen Einschränkungen, die insgesamt Continuations zu einer Angelegenheit mit sehr zweifelhaftem Wert machen. Dabei ist die Bezeichnung „zweifelhaft" ist noch geschönt, wenn man Anwendungsbeispiele[10] betrachtet, die über einen planvollen Rückzug im Fehlerfall hinaus etwas Konstruktives bewirken sollen. Sie sind in ihrer Undurchschaubarkeit mindestens so gefährlich wie Programme mit selbstmodifizierendem Code (das alte Hobby trickreicher Assemblerprogrammierer) oder mit dynamisch modifizierten Operatorpräzedenzen (im alten ALGOL 68 dank Fehldesign möglich).

Auch führen die Einschränkungen nebenbei zu einem erheblichen Dokumentationsaufwand für die Sprache. Damit ein Programmierer überhaupt sicher arbeiten kann, muss ihm ja u. a. mit Begriffen der Programmiersprache vermittelt werden, wie sich das System beim Rückgriff auf eine Continuation genau verhält. Dazu gehören Antworten auf Fragen folgender Art:

- Wird mit den Ursprungswerten weiter gearbeitet?
- Sieht die Sprache vor, dass beim Rückfall auf r im Gegensatz zu einer möglichen Programmierererwartung doch mit dem letzten (neueren) Stand einiger Variablen weitergearbeitet wird.
- Welche Seiteneffekte hat die Nutzung einer Continuation in einem Thread auf den Ablauf anderer Threads?

4 Die JVM unter dem Blickwinkel von Closures und Continuations

Die Java Virtual Machine, kurz JVM, ist ein später Nachfahre der für Pascal realisierten (u. a. im UCSD-Pascal-System eingesetzten) P-Code-Maschine [9]. Wie jene virtuelle Maschine definiert sie in Form ihres Bytecodes zugleich eine Zwischensprache *IL* (für *I*ntermediate *L*anguage) auf dem Weg von einer höheren Programmiersprache hin zum Code einer realen Maschine. Schön ist es, wenn die *IL* wohldefiniert und stabil ist.

[9] Ironischerweise könnte man aber immerhin noch in C++ „heapfrei" nicht nur funktions-, sondern auch objektorientiert arbeiten. Im Normalfall werden dort Objekte auf dem Stack abgelegt. Um analog Java Objekte auf dem Stack abzulegen, muss ja explizit mit Objektzeigern und dem Einsatz des C/C++-**new**-Operators gearbeitet werden.

[10] Z. B. zeigt Pickaxe [15] auf S. 357 einen derartigen sogenannten „Anwendungsfall", in dem der Ruby-Continuation-Caller *callcc* eingesetzt wird - mit der Anmutung „harter Verschlüsselung".

Bereits in den 1950er Jahren (also vor „Urzeiten") wurde im UNCOL-Projekt (UNiversal Computer Oriented Language [3]) versucht, per *IL* die Implementierung von m Programmiersprachen auf n realen Maschinen zu rationalisieren[11]. Das UNCOL-Projekt und seine zahllosen Nachfolger haben sich leider zumeist totgelaufen. Ein Übereinkommen über viele Nutzerkreise hinweg für die „richtige" *IL* ist nicht herstellbar. Eine bestehende *IL* wie die JVM, die aber eine breite Akzeptanz bereits erreicht hat, bietet eine neue Chance, nebenbei den UNCOL-Rationalisierungseffekt zu erreichen. Ein Haken im UNCOL-Sinn liegt allerdings in der sehr starken Bindung der JVM an die Programmiersprache Java. Dank hoher Effizienz, guter Dokumentation und steter Pflege erweckt die JVM gleichwohl hohes Interesse auch javafremder Entwickler. Aus deren Sicht darf eine VM jedoch nicht zu hoch angesiedelt werden. Sie wird dann zu sehr auf ihre „Muttersprache" (also Java im Fall der JVM) zugeschnitten und verliert an Eignung als Zielmaschine für andere Sprachen.

Ein anderer Aspekt ist der Einzug, den die JVM in konkrete Hardwarearchitekturen hält. Es tritt inzwischen die Wirkung ein, die man in der Anfangszeit einmal für Java gesehen hat, nämlich als günstiges Vehikel für die Steuerung von Waschmaschinen, Kaffeeautomaten und ähnlich niederen Geräten (vielleicht Gaspedale an Autos) zu dienen. Beispielsweise bezieht der Hersteller Atmel die JVM für seine 32-Bit-Mikrocontroller ein. Die AVR32-Architektur [1] bietet neben einem eigenen RISC-Befehlssatz bei Nutzung eines bestimmten Erweiterungsmoduls (Java Extension Module) alternativ auch eine Untermenge des Java Bytecode als „in Hardware gegossenen"[12] Befehlssatz an. Allein aus solchen willkommenen JVM-Umsetzungen heraus verbietet sich jedwede Veränderung der Wirkung von Bytecodes, wenn nicht extrem starke Argumente dafür vorliegen. Selbst Erweiterungen des Bytecodes sind problematisch, weil sie JVM-Hardwareumsetzungen veralten lassen. Wenn es einer Spielwiese bedarf, ist stattdessen unbedingt eine Laufzeit-Library vorzuziehen.

Da die JVM eine Stack-Maschine ist, hat sie von vornherein gute Voraussetzungen als Ziel für blockorientierte Programmiersprachen. Tatsächlich gibt es kaum noch eine Programmiersprache, für die kein Compiler mit JVM-Bytecode als Zielsprache existiert. Gleichwohl ist die JVM nicht das Traumziel für FORTRAN oder COBOL angesichts deren fehlender genuiner Blockorientierung und deren starker Betonung von Laufzeiteffizienz. Sie ist auch nicht das Traumziel für C[13] oder C++. Aber der JVM-Bytecode wäre im Prinzip eine brauchbare Zielsprache für dynamisch typisierende Sprachen wie Perl, Python und Ruby. Leider fehlt aus Sicht dieser Sprachen und auch der statisch typisierenden Sprache Scala der JVM eine direkte Unterstützung von Closures und Continuations. Gleichwohl läßt sich trivialerweise die JVM dennoch als Zielmaschine nutzen, wenn man sie analog zu einer in Hardware „gegossenen" CPU mit deren „computational completeness" nutzt.

Bezüglich Closures und Continuations gibt es signifikante Reibungsverluste durch den Zuschnitt der JVM auf ein closure-continuation-freies Java (s. z. B. [4]). Kritisch ist dabei die Frage, wie weit die pure Möglichkeit jener Sprachkonstrukte die Performance der JVM herabsetzt. In diesem Zusammenhang ist die Betrachtung von Ruby interessant. Das, was Ruby in den Versionen 1.8 und 1.9 von der JVM fordert, ist im Endeffekt offensichtlich nicht dramatisch schwer zu realisieren. In seiner Java-Variante JRuby zeigt sich eine Leistungsfähigkeit, die ohne weiteres mit jener anderer Ruby-Interpreter (zumeist in C implementiert) mithalten kann (s. z. B. [2]). Wie andere ist zwar der zitierte diesbezügliche Benchmark-Vergleich im Detail nur mit großer Vorsicht zu genießen, zeigt aber immerhin eine klare Tendenz - nämlich, dass bei hinreichend reduzierter (!) Leistungsfähigkeit von Closures/Continuations die JVM auch für die Nutzung dynamischer Programmiersprachen völlig in Ordnung ist. Die Behauptung, die JVM „sei nicht stark genug" für Ruby etc. ist ziemlich kühn. Trotzdem wurde sie stets (mit Hinweis auf fehlende Closures und Continuations) ins Feld geführt, wenn ein Entwicklerteam eine neue VM in Angriff nahm[14].

Aus Sicht in der Realität auftretender Continuations gibt es offensichtlich keinen Grund, irgendeinen tieferen Eingriff in die JVM vorzunehmen. Die jetzige JVM ist in Wirklichkeit stark genug. Wichtig ist lediglich, dass entsprechende Speicherinformationen (wesentlich dabei ist eigentlich nur der Daten- und der Control-Stack) und Stack-Reset-Befehle nach „außen herausgeführt" werden und damit in einer Laufzeitlibrary aufgenommen und spezifisch für die jeweilige Programmiersprache verarbeitet werden können. Ein Rückzug auf einen alten Zustand r im Sinne der Abbildung 2 ist dann möglich. Bezüglich der „Schmutzeffekte" auf dem Heap und in noch Stackbestandteilen, die

[11] Statt m*n Compiler werden dann nur noch m Compiler mit dem Ziel *IL* und n von *IL* aus die Maschinen versorgende Codegeneratoren (bzw. maschinenspezifische Interpreter) benötigt.

[12] Es geht zwar nur um Mikrocode, von außen gesehen aber gehört der klar zur Hardware.

[13] Hier würde sich auch „die Katze in den Schwanz beißen", da die JVM und die für Java lebensnotwendigen Laufzeit-Libraries massiv von C und der Existenz erstklassiger C-Compiler abhängen.

[14] Zu diesen VMs gehören YARV (Yet Another Ruby VM, s. `http://www.atdot.net/yarv/`) und Parrot (s. `http://www.parrot.org/`).

älter als r sind wird natürlich so „gemogelt wie immer”.

Die Einbeziehung von Closures direkt in die JVM ist kritisch. So etwas stört die normale Stackorganisation erheblich. Einen guten Anhaltspunkt dafür, wie schnell selbst spartanisch einfache Maßnahmen durchschlagen, liefert dabei das von Apple aktuell verfolgte Clang-Projekt[15] mit seinem Konzept von Blockobjekten. Ein Primitivbeispiel (s. `http://lists.cs.uiuc.edu/pipermail/cfe-dev/2008-August/002670.html`) zeigt bereits eine nachhaltige Störung der normalen Stackorganisation:

> „ ... this is a basic idea of Blocks: it is closures for C. It lets you pass around units of computation that can be executed later. For example:
>
> ```
> void call_a_block(void (^blockptr)(int)) {
>   blockptr(4);
> }
> void test() {
>   int X = ...
>   call_a_block(^(int y){ print(X+y); }); // references stack var snapshot
>   call_a_block(^(int y){ print(y*y); });
> }
> ```
>
> In this example, when the first block is formed, it snapshots the value of X into the block and builds a small structure on the stack. Passing the block pointer down to call_a_block passes a pointer to this stack object. Invoking a block (with function call syntax) loads the relevant info out of the struct and calls it. call_a_block can obviously be passed different blocks as long as they have the same type.”

Von einem Compiler sind solche „Schnappschüsse” und deren Nutzung leicht zu managen, auch wenn dies vom „gewöhnlichen” Funktionsaufruf deutlich abweicht. Für einen Einbau in die JVM hingegen bedeutete dies einen weiteren Bytecode-Befehl - mit einer Arbeitsweise, die auf eine ganz bestimmte Vorstellung zugeschnitten ist, wie eine Closure auszusehen hat. Es wäre schon ein ungewöhnlicher Zufall, wenn Clang-Blocktechnik, Funktionsliterale in Scala-Funktionen und Ruby-Closures mit derselben Einkopiertechnik vorbereiteter Funktionsteile auskämen. Nebenbei hätte man eine neue Fehlerquelle für die JVM aufgetan. Ein Blick in die „Innereien” der alles andere als trivialen Struktur der einzukopierenden Bestandteile und deren Nutzung (s. u. a. *gcc/ testsuite/gcc.apple/block-blocks-test-8.c*) würde die „Güte” dieser Fehlerquelle unterstreichen - deren Auflistung sei dem Leser hier erspart.

5 Fazit

Der Papierform nach beherrschen auch außerhalb der LISP-Welt einige Programmiersprachen (insbesondere dynamisch typisierende wie Perl/Python/Ruby) Closures und Continuations. Sie machen daran allerdings aus Performancegründen erhebliche Abstriche. Es handelt sich eher um „Discount-Closures” und „Discount-Continuations”.

Closures sind für theoretische Betrachtungen gelegentlich nützlich. Ihr praktischer Nutzen ist von zweifelhaftem Wert. Von gleichfalls zweifelhaftem Wert sind Continuations. So ist in einer Fehler-Abbruchsituation kaum durchschaubar, welche Werte im Fehlerumfeld geändert sind und welche noch nicht. Der Analogfall von Transaktionen in einem Datenbanksystem zeigt die Gefahren drastisch. Dort wäre es geradezu abenteuerlich riskant, bei TA-Abbruch nicht einwandfrei den Speicherzustand zu Beginn der TA wiederherzustellen[16]. Für manche WEB-Anwendungen wäre hingegen ein nur teilweises Zurücksetzen vielleicht nett. Bei einer Warenbestellung im Internet etwa ist es für den Anwender günstig, wenn bei einem Teilabbruch einer Session ein mühsam aufgebauter Warenkorb seinen Inhalt behält und diesen nur verliert, wenn der Besteller das explizit verlangt.

Der begrenzte Nutzen von Closures und Continuations mit ihrem programmiersprachabhängigen Facettenreichtum lohnt nicht die Integration in eine breit genutzte virtuelle Maschine wie die JVM.

[15] Dessen Ziel ist die Ablösung des in die Jahre gekommenen GNU-C/C++-Systems gcc (GNU Compiler Collection) unter Realisierung eines Laufzeitsystems namens *llvm* (für low level virtual machine)[7].

Bemerkung: Clang möchte gern „cläng” ausgesprochen werden; vielleicht genügt ja auch ein deutsches „Klang”.

[16] Allerdings geht die Liebe zur einwandfreien Wiederherstellung bei DBMS zumeist nicht so weit, innerhalb einer TA gelöschte Tabellen zu rekonstruieren. Die DBMS-Hersteller fürchten den Zorn der Anwender, wenn komplette Tabelle womöglich noch mehrfach zu Undo-Zwecken in Kopie gehalten werden und die Berechnungen der Anwender bezüglich Platz auf den Festplatten völlig über den Haufen werfen.

Deren Umsetzung sollte allein Sache der sprachspezifischen Compiler in Kombination mit sprachspezifischen Libraries sein und nicht die JVM belasten.

Um eine Erweiterung der JVM zu rechtfertigen, müsste von Interessenten schon ein sehr deutlicher Nutzen von Closures und Continuations als notwendige Bestandteile einer Programmiersprache nachgewiesen[17] werden.

Literatur

[1] Atmel 32000B–AVR32–11/07 (AVR32 Architecture Document) http://www.atmel.com/dyn/resources/prod_documents/doc32000.pdf

[2] Cangiano, Antonio: Ruby Benchmarks http://antoniocangiano.com/2008/12/09/the-great-ruby-shootout-december-2008/ (12/2007)

[3] Conway, Melvin E.: "Proposal for an UNCOL", Communications of the ACM 1:3:5 (1958).

[4] Gafter, Neil: Advanced Topics In Programming Languages: Closures For Java http://www.youtube.com/watch?v=0zVizaCOhME (2007).

[5] Goede, Karl: Ist D ein ernstzunehmender Nachfolger für C/C++? - in FINAL, 18. Jahrg., HEft 1, Juni 2008, ISSN 0939-8821, S. 35-59.

[6] http://www.ironruby.net/

[7] Lattner, Chris: Introduction to the LLVM Compiler System http://llvm.org/pubs/2008-10-04-ACAT-LLVM-Intro.pdf

[8] Lindholm, T., Yellin, F.: The Java^{TM} Virtual Machine Specification. Einstiegspunkt http://java.sun.com/docs/books/jvms/second_edition/html/VMSpecTOC.doc.html, (Startjahr: 1999).

[9] Nori, Kesav, Urs Amman; Kathleen Jensen; H. Nägeli; C. Jacobi: Pascal-P implementation notes. In Barron, D. W. (ed.) Pascal: The Language and Implementation. John Wiley & Sons, New York (1991).

[10] Odersky, Martin; Lex Spoon; Bill Venners: Programming in Scala. Artima Press, Mountain View (2008).

[11] Rompf: A Taste of (Scala) 2.8: Continuations. http://www.scala-lang.org/node/2096 (Created by rompf on 2009-06-05. Updated: 2009-06-05, 15:03).

[12] Rose, John R.: Continuations in the VM http://blogs.sun.com/jrose/entry/continuations_in_the_vm (2009).

[13] Sasada, Koichi: Ruby Continuation considered harmful http://www.atdot.net/~ko1/pub/ContinuationFest-ruby.pdf (2008).

[14] Schäfer, Mathias: Performance von JavaScript-Closures http://molily.de/weblog/closures-performance

[15] Thomas, David; A. Hunt: Programmieren mit Ruby. Addison-Wesley (2002).

[17] Ein Nutzen, wie er z. B. für Closures in Javascript geltend gemacht wird [?], ist nur ein sehr zweifelhafter Beleg für deren Notwendigkeit. Closures fungieren dort eher als Ersatzkonstrukt für Funktionen mit Blockkonzept, die Javascript (noch) nicht kennt.

Methoden der Spreadsheet-Entwicklung

Jürgen Jacobs, Phillip Tiburtius

Zusammenfassung: Spreadsheet-Werkzeuge gehören zu den meistgenutzten Softwarewerkzeugen überhaupt. Gleichwohl sind Entwicklungsmethoden für Spreadsheet-Anwendungen nur wenig bekannt. Daher verwundert es nicht, dass fehlerhafte Spreadsheets keine Seltenheit sind. In dieser Arbeit werden ausgewählte Forschungsbeiträge zur Fehlerklassifikation und zur Fehlervermeidung in der Spreadsheet-Entwicklung vorgestellt.

1 Einleitung

Tabellenkalkulationssoftware (engl. Spreadsheet-Software) gehört zu den meistgenutzten Werkzeugen im End-User-Bereich. Nach aktuellen Studien aus den Jahren 2008 [1] und 2009 [2] nimmt die Tabellenkalkulation eine Vorrang-Stellung unter den Business-Intelligence-Werkzeugen ein. Obwohl Spreadsheets in beträchtlichem Ausmaß zur Entscheidungsunterstützung eingesetzt werden, wird die Korrektheit der erstellten Applikationen häufig nicht in Frage gestellt. Dabei können die aus falschen Ergebnissen resultierenden Schäden beträchtlich sein. Ein bekanntes Beispiel ist der Fall eines kanadischen Energieanbieters, dem durch einen cut-and-paste-Fehler ein Verlust von 24 Millionen US-Dollar entstanden ist [3]. In einem weiteren Fall sah sich die US Hypothekenbank Fannie Mae gezwungen, in ihrem Quartalsbericht Korrekturen in Höhe von 1,2 Milliarden US-Dollar vorzunehmen [4][1].

Diese beiden Beispiele sind dabei nicht etwa auf Fehler des Tabellenkalkulationswerkzeugs zurückzuführen, sondern vielmehr die Folge von

[1] Zahlreiche weitere Beispiele sind auf den Seiten der EuSpRIG unter http://www.eusprig.org/stories.htm [Abruf am 9.11.2009] zu finden.

unzulänglichen Entwicklungsprozessen[2]. So sehen zahlreiche Autoren[3] den Grund in einer mangelnden Sensibilisierung von Endbenutzern für die Fehlerentstehung bei der Spreadsheet-Entwicklung. Professionelle Softwareentwickler wissen seit langem, wie fehleranfällig der Entwicklungsprozess ist. Hierin unterscheidet sich der End-User von dem professionellen Softwareentwickler [5]. Eine Sensibilisierung für die Fehlerentstehung allein reicht jedoch nicht aus. Man denke beispielsweise an die Entwicklung eines Spreadsheets mit großen Datenmengen, welches sich über mehrere Arbeitsblätter erstreckt. Selbst erfahrene Benutzer mit einem programmiertechnischen Hintergrund werden in einem solchen Fall Schwierigkeiten bekommen, die korrekte Implementierung aller Formeln der Anwendung zu überblicken. Häufig erhöhen knappe Entwicklungszeiten das Fehlerpotential zusätzlich.

Die Lösungsansätze für das oben beschriebene Problemfeld sind vielfältig. So existieren bspw. softwaregestützte Testwerkzeuge, die sich an datenflussorientierten Testverfahren orientieren oder das Assertion-Konzept als Testinstrument vorschlagen (beide vorgestellt in [5]), Prototypen, die Model-Driven-Engineering (MDE) und objektorientierte Modellierung aufgreifen [6] sowie vielfältige Entwicklungsempfehlungen. Den gesamten Komplex, der sich mit der Thematik der Entwicklung und Nutzung von Spreadsheets befasst, hat Grossman, angelehnt an den 1968 von Bauer geprägten Begriff Software-Engineering, als Spreadsheet-Engineering zusammengefasst [7][4].

In diesem Beitrag wollen wir uns auf Maßnahmen zur Fehlervermeidung konzentrieren. Nach einer Systematisierung möglicher Fehlerarten und -ursachen wird eine Auswahl von möglichen Fehlervermeidungsmaßnahmen in den Phasen Planung, Entwicklung, Kontrolle und Verwendung von Spreadsheet-Applikationen vorgestellt, wobei die Einflussfaktoren Dringlichkeit und Wichtigkeit der Anwendung in die Betrachtung mit einbezogen werden.

2 Fehlerarten und -ursachen

Im Folgenden werden drei unterschiedliche Fehlerklassifikationen vorgestellt, um ausschnittsweise die vielfältigen Sichtweisen auf die Fehlerentstehung aufzuzeigen. Zunächst wird die Klassifikation von Panko und Halverson und deren Überarbeitung

---

[2] Ein Beispiel für softwareseitige Fehler ist bspw. die Fehlinterpretation von Jahrhunderten in MS Excel 97. Siehe hierzu: http://support.microsoft.com/kb/180159/EN-US/ [Abruf am 9.11.2009]

[3] Zugunsten einer besseren Lesbarkeit des Textes haben wir uns dazu entschlossen, auf die Nennung der jeweils weiblichen Form zu verzichten. In der maskulinen Form sind Frauen selbstverständlich mitgemeint.

[4] Im Original: "Spreadsheet engineering is concerned with all aspects of creating spreadsheets."

betrachtet. Die Arbeit von Panko und Halverson wird als der erste ernsthafte Klassifikationsversuch angesehen [8]. Darüber hinaus dienen Teile dieser Klassifikation als Grundlage für viele weitere Ansätze wie z. B. auch den von Rajalingham et al., der näher betrachtet werden soll, weil er als umfangreichster Ansatz zur Klassifizierung von Fehlern gilt. Beide Klassifikationen weisen als Gemeinsamkeit den Versuch auf, den Grund der Fehlerentstehung in die Klassifikation mit einzubeziehen. Dies ist eine Vorgehensweise, die nicht unumstritten ist. So weisen Powell et al. darauf hin, dass der Grund für die Entstehung eines Fehlers im Nachhinein schwer nachweisbar ist, und verzichten daher in ihrer Klassifikation auf die Angabe von Fehlerursachen [8]. Diese andersartige Betrachtungsweise soll daher ebenfalls vorgestellt werden.

2.1 Klassifikation nach Panko und Halverson

Ursprünglich wurde die Klassifikation von Panko und Halverson 1996 als Bestandteil eines multidimensionalen Ansatzes zur Betrachtung von Risiken in der Spreadsheet-Entwicklung veröffentlicht [9]. Im Jahr 2008 wurde die Klassifikation von Panko überarbeitet und die erste Klassifikation zusätzlich erläutert [10]. Die nachfolgende Abbildung zeigt die Version von 1996:

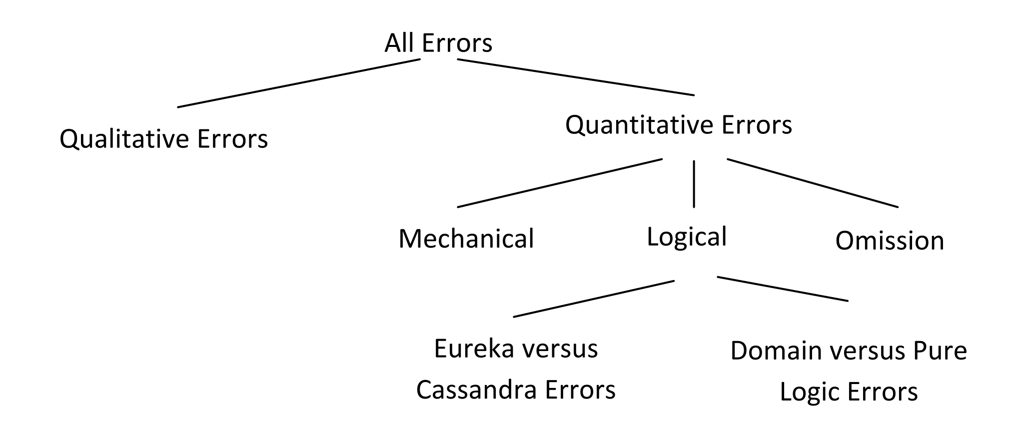


Abb.1 Erste Klassifikation von Panko und Halverson

Auf der obersten Ebene der Klassifikation wird zwischen qualitativen und quantitativen Fehlern unterschieden. Wie von Panko später hinzugefügt wurde, ist das wesentliche Merkmal zur Unterscheidung dieser beiden Klassen der Zeitpunkt der Fehlerentstehung. Quantitative Fehler führen sofort zu falschen Ergebnissen (bottom-line values) im Spreadsheet. Qualitative Fehler dagegen werden von den Autoren als Mängel definiert, die sich negativ auf das Spreadsheetmodell auswirken und im späteren Verlauf zu quantitativen Fehlern führen können (bspw. indem im Rahmen einer Was-wäre-wenn-Analyse Wertänderungen anschließend nicht rückgängig gemacht werden). Quantitative Fehler können in drei Ausprägungen vorkommen. Eine Ausprägung sind mechanische

Fehler, die durch Unachtsamkeiten wie Vertippen oder versehentliches Referenzieren einer falschen Zelle hervorgerufen werden. Mechanische Fehler treten zwar relativ häufig auf, werden aber meist frühzeitig entdeckt und korrigiert. Eine weitere Ausprägung der quantitativen Fehler bezieht sich auf die Logik der im Spreadsheet verwendeten Formeln. Hier wird auf der einen Seite unterschieden, ob der Fehler leicht (Eureka Errors) oder schwer (Cassandra Errors) nachweisbar ist. Auf der anderen Seite wird unterschieden, ob es sich um einen reinen Logikfehler handelt oder ob der Fehler auf ein falsches Verständnis des Anwendungsbereichs zurückzuführen ist („die richtige Umsetzung der falschen Formel“). Die Aufdeckungsrate bei logischen Fehlern ist nach Panko geringer als bei den mechanischen Fehlern. Mit Omission Errors – der dritten Klasse der quantitativen Fehler – werden Mängel im Spreadsheetmodell bezeichnet. Die Aufdeckungsrate dieser Fehlerklasse wird von den Autoren als sehr gering eingestuft.

Die erste Klassifikation von Panko und Halverson muss durchaus kritisch betrachtet werden. Zum einen ist die Klassifikation nicht konsistent. So wird bspw. nicht geklärt, weshalb die Klassen der Pure und Domain Logic Erros nicht bezüglich ihrer Nachweisbarkeit unterschieden werden. Daneben sind nur wenige Klassen mit konkreten Beispielen unterlegt, wodurch das Verständnis der ohnehin nicht immer präzisen Definitionen erschwert wird. Durch die nachträgliche Anmerkung von Panko zur Unterscheidung der qualitativen und quantitativen Klasse wurde zwar mehr Klarheit geschaffen, dennoch würde eine weitere Unterteilung der qualitativen Fehler die Abgrenzung erleichtern.

Die von Panko überarbeitete Version hat einige grundlegende Änderungen im Aufbau gegenüber der Vorversion erfahren. Vor der Einteilung in qualitative und quantitative Fehler wird nun auf der höchsten Ebene unterschieden, ob ein Fehler mutwillig oder unfreiwillig verursacht wurde. Eine Erweiterung der Klasse der qualitativen Fehler erfolgt jedoch auch in der überarbeiteten Version nicht. Die Aufteilung unterhalb der Klasse der quantitativen Fehler wurde völlig neu gestaltet. Dabei wird nun unterschieden, ob der Fehler auf eine falsche Intention zurückzuführen ist oder ob die richtige Intention falsch umgesetzt wurde (z. B. durch Vertippen). Die Klasse der Omission Errors ist nun nicht mehr eigener Bestandteil der Klassifikation, sondern ist der Klasse der Context Errors zuzuordnen. Mechanische Fehler werden nun als Slips bezeichnet. Auch in der überarbeiteten Version fehlen eine genaue Abgrenzung der Klassen und aussagekräftige Beispiele. Die nachfolgende Abbildung zeigt die überarbeitete Version der Klassifikation:

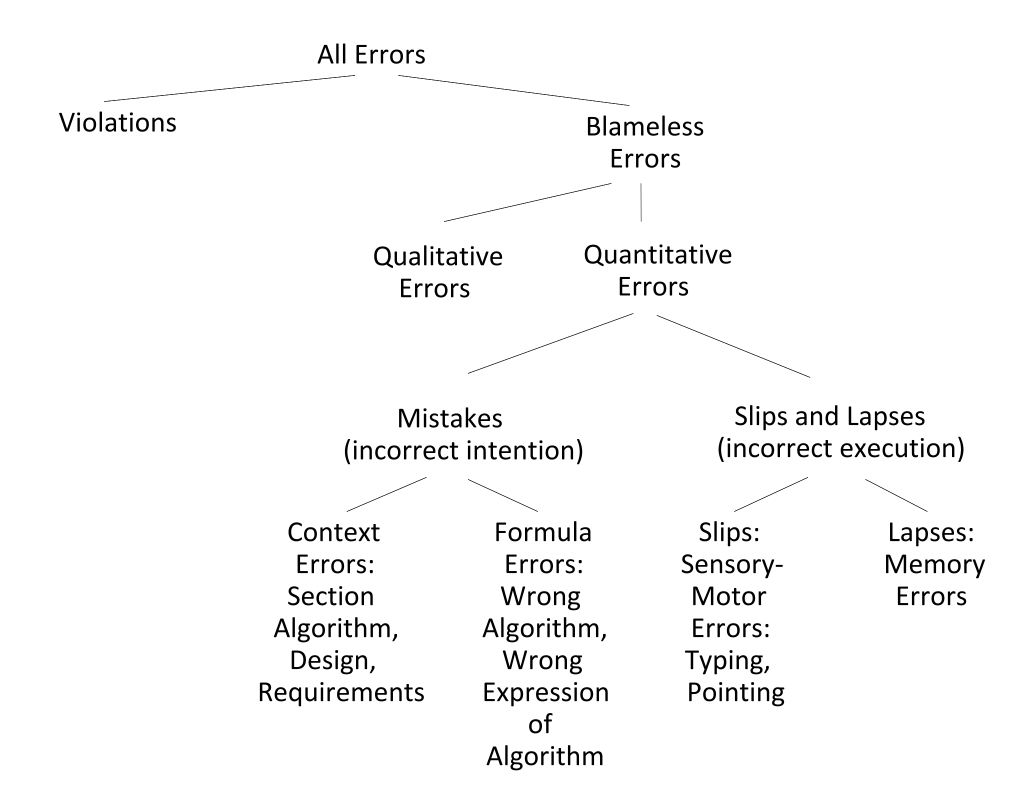


Abb.2 Überarbeite Version der Klassifikation von Panko und Halverson

Eine ebenfalls von Panko erstellte Zusammenfassung von Studien über die Fehlerentstehung kommt zu dem Ergebnis, dass von 113 betrachteten Spreadsheets 88 % Fehler beinhalten. Dem Autor zufolge sind selbst bei sorgfältig entwickelten Spreadsheet-Anwendungen mindestens ein Prozent aller Zellen, die Formeln beinhalten, fehlerhaft. Neben der Auswertung bereits vorliegender Studien hat der Autor 2003 zwei Spreadsheet-Prüfer aus Großbritannien interviewt[5]. Beide Prüfer haben pro Jahr im Schnitt 36 Spreadsheets untersucht. Es stellte sich dabei heraus, dass kein Spreadsheet fehlerfrei war. Beide Prüfer gaben an, dass in 5 % der Fälle schwerwiegende Fehler vorlagen, die weitreichende Auswirkungen gehabt hätten, falls sie unentdeckt geblieben wären [11].

2.2 Klassifikation nach Rajalingham et al.

Die Arbeit von Rajalingham et al. berücksichtigt auf der höchsten Ebene Softwarefehler des Tabellenkalkulationswerkzeugs [12]. Bei der Abgrenzung zwischen qualitativen und quantitativen Fehlern orientieren sich Rajalingham et al. zwar an der Definition nach Panko und Halverson, die feinere Aufteilung ihrer Klassifikation ermöglicht es jedoch, die Klasse der qualitativen Fehler auf die Bereiche Semantik und Wartung einzugrenzen.

[5] Unter bestimmten Bedingungen ist in Großbritannien die Kontrolle von Spreadsheets vorgeschrieben.

Die folgende Abbildung zeigt den oberen Bereich der Klassifikation von Rajalingham et al.[6]:

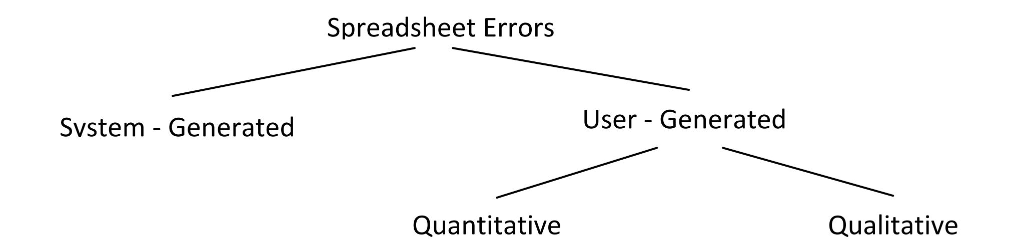


Abb.3 System- und benutzergenerierte Fehler

Quantitative Fehler entstehen nach Rajalingham et al. entweder versehentlich (Accidental Errors) oder aufgrund falscher Schlussfolgerungen (Errors in Reasoning). Erstgenannte orientieren sich an der Definition von mechanischen Fehlern nach Panko und Halverson. Es handelt sich um kleine Unachtsamkeiten, die häufig sofort vom Verursacher bemerkt werden. Die folgende Abbildung zeigt die Einteilung der quantitativen Fehler:

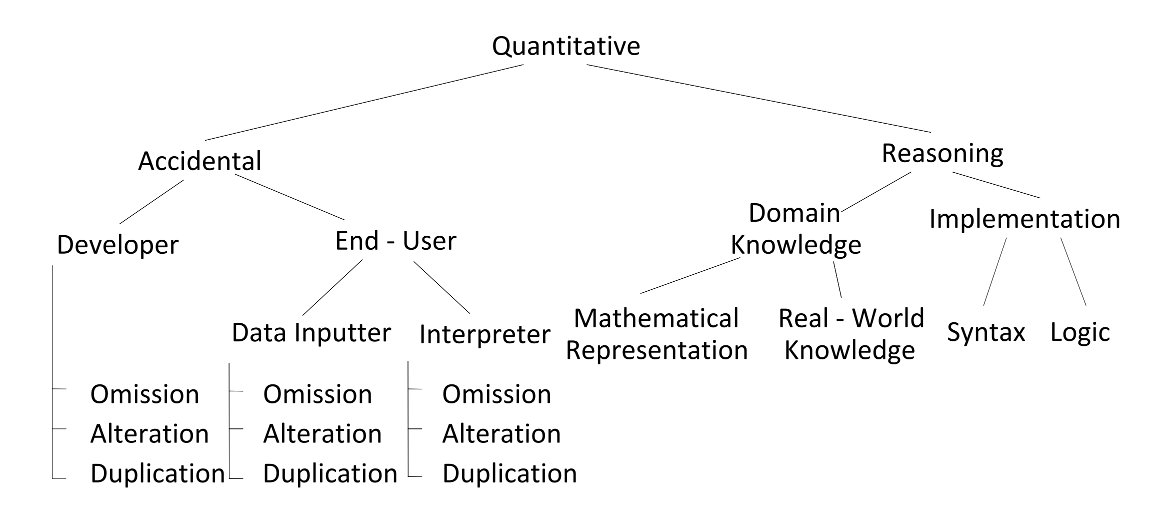


Abb.4 Quantitative Fehlerklassen nach Rajalingham et al.

Die Autoren unterscheiden hinsichtlich der Ausprägungen der Klasse der Accidental Errors zwischen der Entwickler- und der Benutzerseite. Für die Beschreibung wird ein Rollenmodell verwendet, in dem zwischen Entwickler und im End-User-Bereich zwischen dem Bereitsteller der Daten und dem Empfänger der Daten unterschieden wird. Grundlage bilden dabei Fehlerklassen, die sich auf Versäumnisse, Modifikationen und Kopiervorgänge beziehen. In Tabelle 1 sind die verschiedenen Ausprägungen dieser Fehlerklassen näher erläutert:

[6] Für eine bessere Übersicht werden die einzelnen Oberklassen der Klassifikation separat betrachtet.

Rolle	Fehlerart		
	Omission Error	Alteration Error	Duplication Error
Spreadsheet-Entwickler (Developer)	Versäumnisse bei der Erstellung des Spreadsheet-Modells, bspw. fehlende Referenzen auf Eingabezellen	Änderung an bereits bestehenden Modellen, sodass ein fehlerhaftes Verhalten der Anwendung entsteht – bspw. durch versehentliche Aktivierung des Zellschutzes	Elemente des Spreadsheet-Modells werden unbeabsichtigt mehrfach verwendet, bspw. die mehrfache Definition einer Variablen
Bereitsteller der Daten (Data Inputter)	Auslassungen von Daten, die vom Spreadsheet-Modell benötigt werden	Bspw. wird eine Summenformel nach dem Hinzufügen zusätzlicher Zeilen nicht angepasst	Bspw. versehentliches Einfügen kopierter Daten an der falschen Stelle
Empfänger der Daten (Interpreter)	Unvollständige Betrachtung der Daten	Bspw. werden von einer Reihe von zusammenhängenden Werten nur einige sortiert	

Tab.1 Rollenmodell nach Rajalingham et al.

Während auf der einen Seite quantitative Fehler durch Unachtsamkeit verursacht werden können, ist auf der anderen Seite die Klasse der Fehler zu betrachten, die durch falsche Schlussfolgerungen entstehen. Diese beziehen sich entweder auf die Domäne des Spreadsheet-Modells oder die Implementierung. Unzureichendes Domänenwissen bezieht sich entweder auf eine Wissenslücke bezüglich des zu modellierenden Sachverhalts selbst (Real-World Knowledge) oder auf fehlende Kenntnisse in der mathematischen Darstellung des Sachverhalts (Mathematical Representation). Ein Beispiel für die erstgenannte Klasse ist die Verwechslung von linearer und degressiver Abschreibung. Beispielhaft für die zweite Klasse ist die fehlerhafte Umsetzung von Verhältnisgleichungen in der Prozentrechnung. Hinsichtlich der Implementierung wird unterschieden, ob syntaktische oder logische Fehler vorliegen. Syntaktische Fehler entstehen, wenn bei der Erstellung des Spreadsheets Ausdrücke verwendet werden, die der Tabellenkalkulationssoftware unbekannt sind. Diese Fehler sind jedoch relativ harmlos, da sie i. d. R. von der Software gemeldet werden. Logische Fehler entstehen, wenn der Benutzer eine Funktion der Software falsch interpretiert, z. B. durch die Verwechslung absoluter und relativer Referenzen.

Wie Abbildung 5 zeigt, unterscheidet sich die Klasse der qualitativen Fehler hinsichtlich der Semantik und der Wartung:

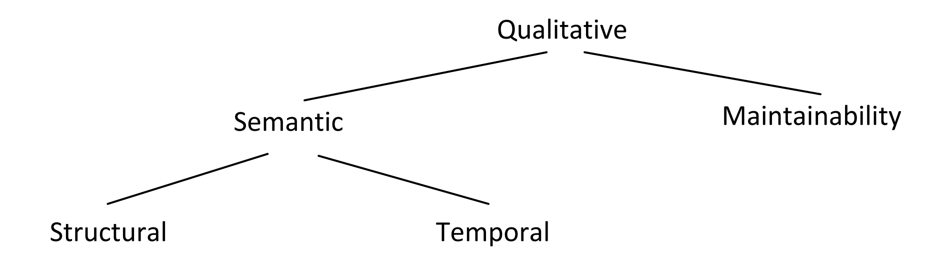


Abb.5 Qualitative Fehlerklassen nach Rajalingham et al.

Semantische Fehler sind dadurch charakterisiert, dass der Benutzer als Folge von Unklarheiten über die Bedeutung von im Spreadsheet verwendeten Daten falsche Schlüsse zieht. Rajalingham et al. charakterisieren diese Fehlerart als nur schwer entdeckbar. Derartige Fehler entstehen entweder durch Mängel im Aufbau des Spreadsheet-Modells (Structural Error) oder durch veraltete Daten (Temporal Error, z.B. bei Währungsumrechnungen). Ein Beispiel für die erschwerte Wartung ist die Verwendung von Konstanten anstelle von Referenzen in Formeln (hard-coding). Hard-coding führt zu fehlerhaften Formeln, falls die Aktualisierung einer Konstanten nicht an allen Stellen ihrer Verwendung vorgenommen wird.

2.3 Klassifikation nach Powell et al.

Powell et al. vertreten die Meinung, dass nicht die Anzahl der Fehler bedeutsam ist, sondern die (quantitativen) Auswirkungen eines Fehlers auf die Ergebnisse im Spreadsheet [13]. Im Rahmen ihrer Studie wurden jeweils fünf Spreadsheets von zwei Beratungsunternehmen, einem großen Finanzdienstleister, einem produzierenden Unternehmen und einer Bildungseinrichtung untersucht. Insgesamt wurden in den 25 Spreadsheets 381 potenzielle Fehler entdeckt, von denen 117 nach Rücksprache mit den Spreadsheet-Entwicklern tatsächlich als Fehler eingestuft wurden. Von den 117 Fehlern hatten 70 quantitativen Einfluss (ein Fehler verfälschte das Ergebnis um 100 Millionen Dollar).

Powell et al. gewinnen aus ihren Untersuchungen folgende Erkenntnisse:

- In einigen Organisationen gab es Spreadsheets, die im Wesentlichen frei von Fehlern waren.
- Der Kenntnisstand der Mitarbeiter über den Umgang mit Spreadsheets reichte innerhalb der einzelnen Organisation von hervorragend bis schwach.
- In einigen Organisationen waren Spreadsheets mit einer hohen Anzahl an Fehlern im Einsatz. Einige dieser Fehler hatten beträchtliches Ausmaß.
- Ein Großteil der Fehler hatte keinen Einfluss auf das Ergebnis oder wirkte sich nur auf zu vernachlässigende Berechnungen aus.

- Der Zusammenhang zwischen der Bedeutung der Ergebnisse des Spreadsheets für die Organisation und ihrer Qualität ist gering.
- Aus der Sicht der Spreadsheet-Entwickler beinhalten nur wenige Spreadsheets Fehler, welche die tatsächliche Verwendbarkeit einschränken.

Die Entwickler wurden von den Autoren zusätzlich befragt, welche Gründe ihrer Meinung nach für qualitativ minderwertige Anwendungen vorliegen. Zum einen wurde der Zeitdruck genannt, unter dem viele Spreadsheets erstellt werden müssen. Zum anderen wurde ein „organisches Design“ bemängelt. Das bedeutet, dass betroffene Spreadsheets entweder eine Abwandlung eines vorausgegangenen Spreadsheets waren oder dass das Spreadsheet während der Entwicklungsphase ohne ausreichende Planung „gewachsen“ ist. In diesem Zusammenhang wurden auch die wechselnden Anforderungen während der Entwicklung kritisiert. Ein weiterer Grund für die Fehlerentstehung sind nicht vorhandene oder unzureichende Testphasen, sodass in den meisten Fällen die Spreadsheets nicht überprüft wurden. Mangelndes Wissen über die Erstellung von Spreadsheets wurde bemerkenswert selten als Grund angegeben (aufgezeigte Fehler in den Formeln wurden von den Testpersonen relativ schnell entdeckt). An den hier genannten Sachverhalten, also unzureichender Planung, knappe Entwicklungszeiten und unzureichende Kontrolle setzen die in Kapitel 3 vorgestellten Maßnahmen an.

Wie eingangs bereits erwähnt wurde, verzichten die Autoren auf den Versuch, die Entstehungsursachen von Fehlern für eine Klassifikation zu verwenden, da diese Ursachen allenfalls in Laboruntersuchungen durch Beobachtung oder Befragung der Entwickler herauszufinden wären, nicht aber durch eine reine Analyse des erstellten Spreadsheets. So kann beispielweise eine fehlerhafte Finanzformel viele Ursachen haben: Tippfehler, fehlendes Fachwissen, fehlende Kenntnisse über die Wirkung der Formel oder ein unvollständiges Modell [8]. Beobachtbar ist allein ein logischer Fehler in Form einer fehlerhaften Finanzformel. Basierend auf Auswertungen von Audits schlagen die Autoren eine Taxonomie von 6 Kategorien vor, die eine weitgehend zuverlässige Zuordnung beobachteter Fehler ermöglichen [14] und [15]:

Fehlerkategorie	Fehlerbeschreibung
Logic error	Inkorrekte Verwendung einer Formel führt zu einem falschen Ergebnis
Reference error	Eine Formel enthält einen oder mehrere fehlerhafte Zellverweise
Hard-coding	Verwendung von Konstanten in Formeln verschlechtert die Wartbarkeit
Copy/Paste Error	Inkorrekt ausgeführte Copy/Paste Aktionen führen zu fehlerhaften Formeln
Data input error	Verwendung fehlerhafter Eingabedaten (bezogen auf die Eingabezellen)
Omission error	Werte von Eingabezellen für Formeln werden leer gelassen

Tab.2 Fehlerarten nach Powell et al.

Die in den einzelnen Kategorien aufgetretenen Fehlerhäufigkeiten (basierend auf 50 Audits) sind in Tabelle 3 aufgeführt:

Fehlerkategorie	Anzahl Fehler	Anzahl fehlerhafter Zellen
Hard-coding	182 (37,7%)	2111 (43,5%)
Reference error	159 (32,9%)	1074 (22,1%)
Logic error	106 (21,9%)	1389 (28,6%)
Copy/Paste Error	17 (3,4%)	206 (4,2%)
Omission error	13 (2,7%)	65 (1,3%)
Data input error	6 (1,2%)	10 (0,2%)
Total	483	4855

Tab.3 Fehlerhäufigkeiten nach Powell et al.

Neben den hier vorgestellten Klassifikationen gibt es noch zahlreiche weitere [8]. Insgesamt können die bis dato existierenden Klassifikationen nicht als vollständig bzw. als ausgereift angesehen werden. Die vorliegenden Klassifikationen werden von Powell et al. aus folgenden Gründen als unzureichend kritisiert [13]:

- Klassifikationen werden ohne Angabe des Zwecks ihrer Erstellung veröffentlicht.
- Bestehende Klassifikationen unterlegen die Fehlerarten nicht ausreichend mit Beispielen.
- Vorhandene Klassifikationen sind nicht ausreichend getestet, sodass Inkonsistenzen auftreten können.
- Die Abgrenzung von qualitativen und quantitativen Fehlern ist ungenau.

Dem sei noch hinzugefügt, dass häufig die Methode, nach der die Klassifikation erstellt wurde, nicht ausreichend erläutert wird und dass nur unzureichend auf die Verbreitung der beschriebenen Fehlerarten eingegangen wird.

3 Maßnahmen zur Fehlervermeidung

Ebenso wie die Fehlerklassifikationen sind auch die vorgeschlagenen Lösungsansätze zur Fehlervermeidung vielfältig. Zum einen gibt es Entwicklungswerkzeuge – beispielsweise zur Testunterstützung oder zur Unterstützung des objektorientierten Model-Driven-Engineering-Konzepts. Bis auf kommerzielle Auditing-Werkzeuge sind diese Ansätze bisher allerdings nur prototypisch implementiert worden. Zum anderen findet man zahlreiche Empfehlungen – beispielsweise zur Anordnung der Informationen im Spreadsheet, der Implementierung von Formeln und zum Einsatz von Systemfunktionen zur Fehlervermeidung. Sowohl der Werkzeugeinsatz als auch die Empfehlungen können in Entwicklungsmodelle integriert werden, die oftmals den Anspruch haben, den gesamten Lebenszyklus des Spreadsheets von der Planung bis hin zur Archivierung abzudecken.

Unser Beitrag orientiert sich an dem Lebenszyklusmodell von Read und Batson [16]. Als erstes wird die Planungsphase betrachtet. In dieser Phase erfolgt die Beschreibung der Problemstellung, die durch das Spreadsheet modelliert werden soll. Dazu sind Fragen zur

Datenbeschaffung, den durchzuführenden Berechnungen aber auch der Bewertung von Alternativen zum Einsatz von Tabellenkalkulationssoftware zu beantworten. Auf der Grundlage der Ergebnisse der Planungsphase kann das Design bzw. die Entwicklung erfolgen. In der Design- und Entwicklungsphase liegt der Fokus auf der Anordnung von Daten und Ergebnissen, der Implementierung der Formeln, sowie der Konsolidierung bei Verwendung von mehreren Arbeitsblättern. Nachdem die Entwicklung abgeschlossen ist, wird in der Testphase überprüft, ob das Spreadsheet die Anforderungen der Planung erfüllt und korrekte Ergebnisse liefert. Hierzu wird beispielhaft der Einsatz eines Testwerkzeuges dargestellt. Da Fehler nicht nur in der Entwicklungsphase sondern auch während der Verwendung bzw. durch nachträgliche Modifikationen entstehen können, wird auf diese Phase im Lebenszyklus des Spreadsheets ebenfalls eingegangen.

Der große Vorteil der Tabellenkalkulation liegt in ihrer Flexibilität. Bei der Ausarbeitung von Entwicklungsmodellen muss beachtet werden, dass die flexible Einsetzbarkeit von Spreadsheets nicht eingeschränkt wird. Read und Batson tragen dieser Problematik Rechnung, indem sie den Umfang der vorgestellten Maßnahmen in ihrem Lebenszyklusmodell an verschiedenen Rahmenbedingungen, wie bspw. der zur Verfügung stehenden Entwicklungszeit oder der Komplexität der zu modellierenden Problemstellung, ausrichten. Madahar et al. entwickeln dimensionsorientierte Modelle, um Kategorien für die Spreadsheet-Verwendung und dem damit verbundenen Einsatzrisiko zu gewinnen [17]. Eine Alternative zu dieser dimensionsorientierten Ausrichtung ist eine Betrachtungsweise, bei der das empfohlene Modell vom Typ des Spreadsheet-Entwicklers (und den damit verbundenen Abstufungen an Vorkenntnissen in der Anwendungsdomäne einerseits und der Softwareentwicklung andererseits) abhängig gemacht wird [7]. Jedoch existieren zurzeit noch keine Konkretisierungen dieses Ansatzes. Wir folgen daher dem dimensionsorientierten Ansatz, um Einflussfaktoren für die Spreadsheet-Entwicklung aufzuzeigen. Um den vielfältigen Anwendungsbereichen der Tabellenkalkulation Rechnung zu tragen, wird im Gegensatz zu Madahar et al. jedoch von einer Typisierung des Verwendungszwecks abgesehen. Stattdessen wird die Struktur des Spreadsheets als Dimension herangezogen.

Aus der Problemstellung, welche durch das Spreadsheet modelliert werden soll, ergibt sich mit den benötigten Daten und deren Verknüpfung in Form von Formeln bzw. Berechnungen (im weiteren Sinne auch Makros etc.) die Struktur des Spreadsheets. Je komplexer die Struktur ist, desto größer sind der Entwicklungsaufwand und das Risiko, dass Informationen, die dem Spreadsheet entnommen werden, fehlerhaft sind. Je kürzer die Zeitspanne ist, die für die Entwicklung zur Verfügung steht, desto größer ist die Wahrscheinlichkeit, dass Fehler entstehen und übersehen werden, da Planungs- und Testphasen aufgrund der Zeitnot ausbleiben oder unzureichend sind. Neben der Struktur und der Dringlichkeit ist auch Bedeutung des Spreadsheets ein zu berücksichtigender Faktor. Die Ausprägungen aller drei Dimensionen bestimmen die Kategorie der zu entwickelnden Anwendung. Diese hat Auswirkungen auf den Aufwand, der die Breite und Tiefe der Maßnahmen zur Fehlervermeidung und Fehlerfindung bestimmt sowie auf das Risiko der Fehlerentstehung. Je geringer der betriebene Aufwand ist, desto höher wird

das Fehlerrisiko sein, welches in Kauf genommen werden muss. In Abbildung 6 wird dieser Zusammenhang noch einmal veranschaulicht:

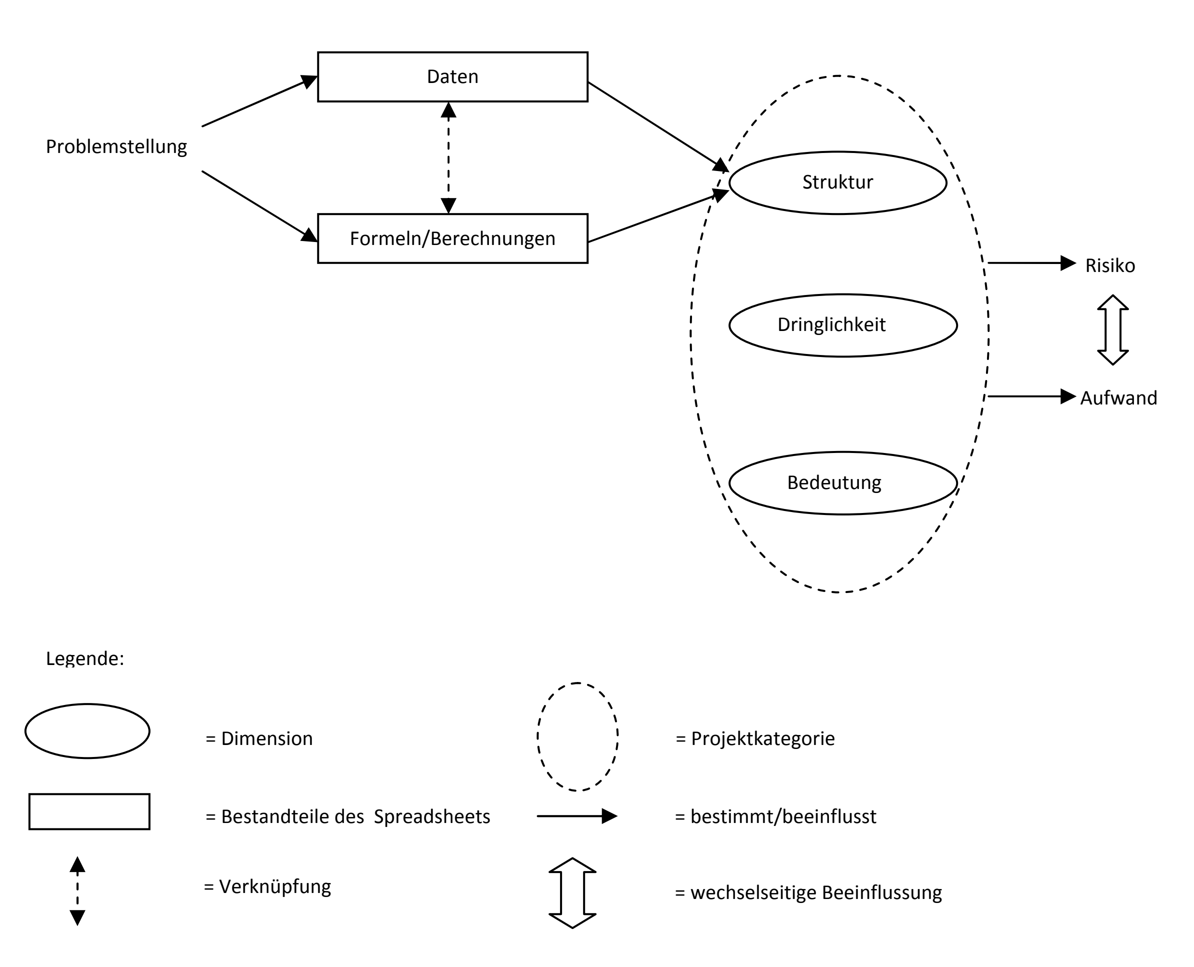


Abb.6 Einflussfaktoren in der Spreadsheet-Entwicklung

Ein Beispiel für eine simple Spreadsheet-Struktur ist die Verwendung des Spreadsheets als Datenspeicher bzw. als Datenquelle. In diesem Fall steht ein möglicherweise großes Datenvolumen einer geringen Menge von Formeln (bspw. nur Aggregatfunktionen) gegenüber. Als mögliche Maßnahmen mögen hier eine Kontrolle der Datenintegrität und die Aktivierung des Zellschutzes zur Vermeidung unbeabsichtigter Änderungen reichen. Im Gegensatz dazu zeichnet sich eine komplexe Spreadsheet-Struktur durch ein großes Datenvolumen aus, auf dessen Grundlage eine große Menge an (komplexen) Berechnungen durchgeführt wird. Ein Beispiel hierfür ist die Finanzplanung eines Unternehmens. Da die Ergebnisse dieser Anwendung in aller Regel von großer Bedeutung für das Unternehmen sein werden, ist möglichst der volle Umfang der Maßnahmen auszuschöpfen, um die Entstehung von Fehlern zu vermeiden. Neben einer umfangreichen Planungsphase (Roadmap, detaillierte Beschreibung von Daten, Berechnungen und Ergebnissen etc.) sind während der Entwicklung immer wieder die Ergebnisse zu überprüfen. In der Kontrollphase werden die Erstellung eines detaillierten Testplans, der Einsatz von Testwerkzeugen und die Überprüfung durch Dritte erforderlich sein. Darüber hinaus ist die Anwendung zu dokumentieren (sowohl durch eine Anleitung

zur Benutzung des Spreadsheets, als auch durch eine technische Dokumentation sowie ein Changelog). Für ein solches Vorhaben sollte die entsprechende Entwicklungszeit zur Verfügung gestellt werden.

3.1 Planung

In der Planungsphase werden grundlegende Überlegungen zur Entwicklung des Spreadsheets angestellt. In Abschnitt 2.3 wurden unzureichende Planungsmaßnahmen als einer der Hauptgründe für Qualitätsprobleme der Anwendung genannt (auch indirekt durch das ungeplante Wachsen der Anwendung während der Entwicklung). In einer von Powell et al. durchgeführten Befragung wurde festgestellt, dass mehr als die Hälfte der Unternehmen – sog. HA (high awarenes)-Unternehmen – ihre Spreadsheet-Entwicklung mit einem ausgeprägten Bewusstsein für Fehlerentstehung planen [18]. Eine sehr umfassende Planungsphase wird in dem Lebenszyklusmodell von Read und Batson [16] beschrieben. Diese soll im Folgenden verkürzt dargestellt werden:

Die erste Stufe der Planungsphase dient der Festlegung von Zielen, d. h. es ist zu definieren, welche Problemstellung durch das Spreadsheet modelliert werden soll und welche Ergebnisse produziert werden sollen. Dies schließt auch die Festlegung der Grenzen der Anwendung mit ein. Darüber hinaus sind Anforderungen bzgl. der Datenbeschaffung und der personellen Ausstattung des Projekts zu klären. Nach der Projektkategorie – also Struktur, Dringlichkeit und Bedeutung – richtet sich das weitere Vorgehen. Für die Planungsphase selbst bedeutet dies, dass mit zunehmender Komplexität der Struktur auch der Umfang der Planungsphase steigen sollte (durch die Erstellung von Zeitplänen, detailliertere Beschreibung von Berechnungen etc.). Auch bei zeitkritischen Projekten sollte nicht gänzlich auf eine Planungsphase verzichtet werden, insbesondere nicht bei Projekten von großer Bedeutung. Stattdessen ist es sinnvoll, den Umfang der Planung entsprechend zu reduzieren, bspw. durch den Verzicht auf umfangreiche Roadmaps und die Konzentration auf essentielle Teile der technischen Dokumentation. Nach der ersten Stufe der Planungsphase kann die Spezifikation von Daten, Berechnungen und Ergebnissen erfolgen. Durch die Spezifikation der Berechnungen können logische Fehler nach Panko und Halverson (bzw. der Reasoning-Errors nach Rajalingham et al. und Logic-Errors nach Powell et al.) vermieden werden. Die Ergebnisse stellen den wichtigsten Bestandteil eines Spreadsheets dar. Dementsprechend ist es empfehlenswert, dass der Spezifikationsprozess entgegengesetzt zum Logikfluss erfolgt, also von den Ergebnissen über die Berechnungen hin zu den Eingabedaten. Die folgende Abbildung verdeutlicht dieses Vorgehen:

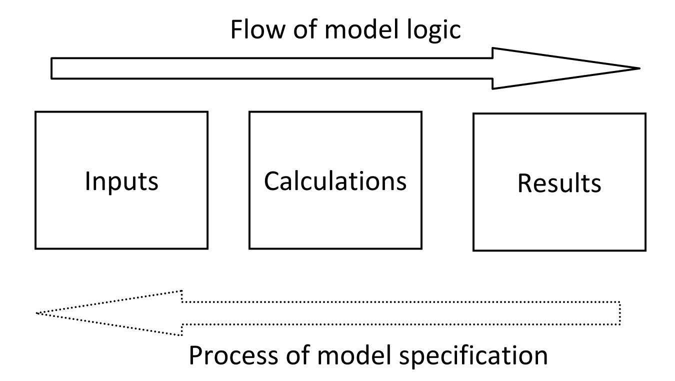


Abb.7 Spezifikationsphase nach Read und Batson

Nach der Festlegung der Ergebnisse und der Beschreibung, auf welchen Daten und Berechnungen sie beruhen, sollten alternative Werkzeuge wie weitere Business-Intelligence-Systeme, Datenbanken oder der Einsatz von Programmiersprachen geprüft werden, um festzustellen, ob diese nicht unter Umständen besser zur Lösung der Problemstellung geeignet sind [19].

3.2 Design und Entwicklung

Laut Panko ist mangelhaftes Spreadsheet-Design ein wesentlicher Grund für die Fehlerentstehung [11]. Der Aufbau und die Funktionsweise von schlecht designten Spreadsheets sind für Dritte in den nachgelagerten Stufen des Lebenszyklus nur schwer nachvollziehbar. Wartung und Modifikation werden dadurch erschwert. Auch für Entwickler kann mangelhaftes Design eine Fehlerquelle darstellen. Analog zum Software-Engineering gibt es für das Spreadsheet-Engineering unterschiedliche Ansichten über eine geeignete Anordnung von Daten, Funktionen und Ergebnissen.

Der klassische Ansatz empfiehlt eine Trennung von Daten, Berechnungen und Ergebnissen. Dieser modulare Ansatz wird unter anderem von Read und Batson [16] vorgeschlagen. Bezogen auf die Tabellenkalkulation bedeutet dies, dass die Eingabedaten, die Berechnungen und die Ergebnisse in separaten Bereichen gehalten werden (eine Praxis, der auch der Großteil der HA-Unternehmen folgt). Diese Bereiche können durch räumliche Trennung innerhalb eines Arbeitsblatts, durch mehrere Arbeitsblätter in einem Spreadsheet oder sogar mehrere Spreadsheets (Arbeitsmappen) realisiert werden. Die räumliche Trennung kann durch den Einsatz von verschiedenen Farben für die einzelnen Bereiche und den Einsatz unterschiedlicher Schriftarten und Formatierungen unterstützt werden.

Das folgende Beispiel einer Umsatzplanung zeigt eine Unterteilung auf einem einzelnen Arbeitsblatt (vgl. Abbildung 8). Die Daten sind dabei Grün unterlegt. Berechnungen werden in Orange dargestellt und das endgültige Ergebnis in Lila. Man kann jedoch schon an diesem einfachen Beispiel anhand der Zwischenergebnisse in den Zeilen 8, 15, 18 und

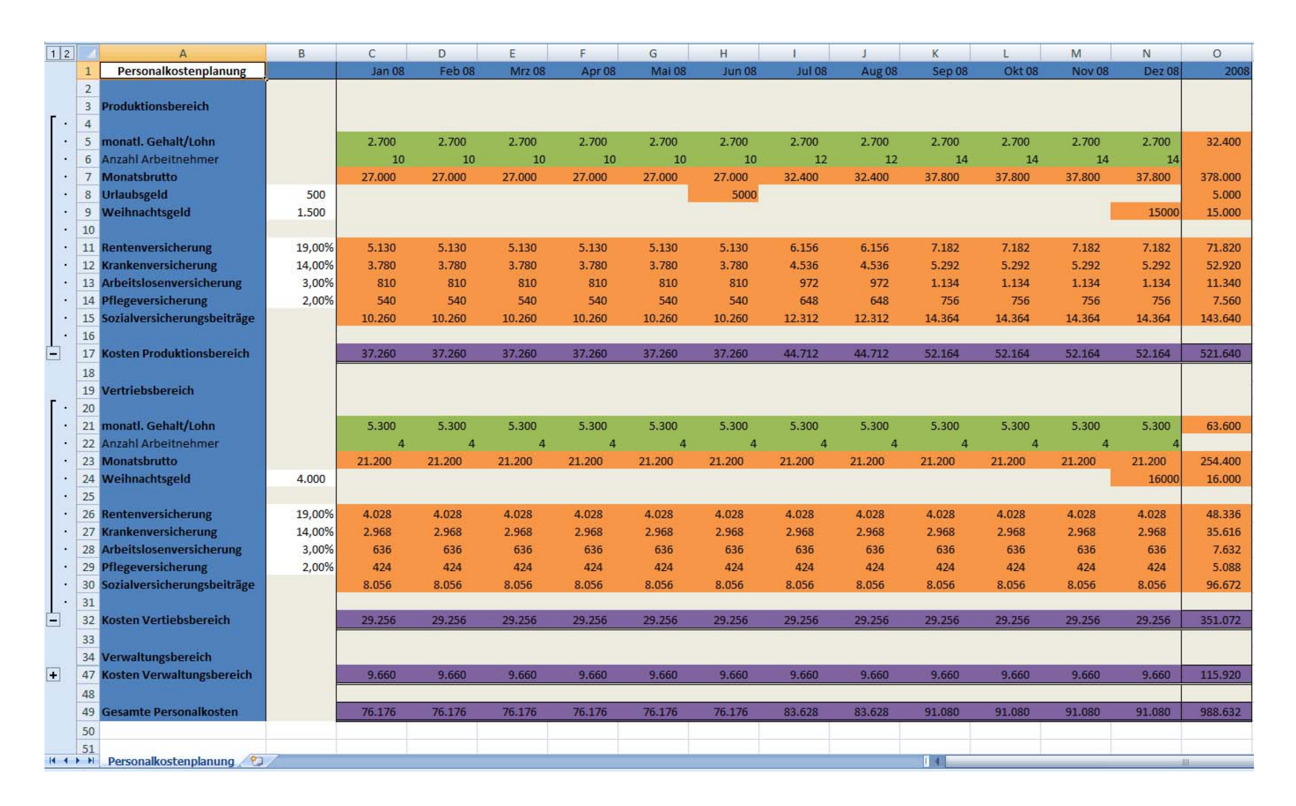
den Jahresergebnissen erkennen, dass eine allzu strikte Trennung (etwa durch Zusammenlegung der drei grünen Bereiche) nicht immer sinnvoll bzw. umsetzbar ist.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	AB	AG
1	Umsätze in €		Jan 07	Feb 07	Mrz 07	Apr 07	Mai 07	Jun 07	Jul 07	Aug 07	Sep 07	Okt 07	Nov 07	Dez 07	2007	2008	2009
2																	
3	Gruppe I																
4	Produkt A		20.000	20.000	25.000	27.000	27.500	27.500	27.000	30.000	29.000	29.000	29.100	31.000	322.100	398.500	468.700
5	Produkt B		80.000	85.000	89.000	92.000	91.000	91.000	92.000	95.000	95.000	95.000	93.000	93.000	1.091.000	1.233.000	1.589.000
6	Produkt C		25.000	25.000	24.000	24.500	25.000	25.000	24.500	24.500	24.500	24.500	24.000	25.000	295.500	310.000	387.900
7	Produkt D		30.000	30.000	31.000	34.000	34.000	34.000	34.000	34.500	34.500	34.500	34.000	35.000	399.500	423.000	467.400
8	Gesamt Gruppe I		155.000	160.000	169.000	177.500	177.500	177.500	177.500	184.000	183.000	183.000	180.100	184.000	2.108.100	2.364.500	2.913.000
9																	
10	Gruppe II																
11	Produkt E		9.000	9.100	9.200	8.900	9.000	9.400	9.300	9.200	9.000	9.000	9.300	9.100	109.500	121.000	125.700
12	Produkt F		8.500	8.500	8.500	8.000	8.100	9.000	9.100	9.000	8.900	8.900	9.200	9.200	104.900	110.600	134.000
13	Produkt G		21.300	22.000	22.100	19.000	19.500	21.000	21.400	22.000	21.900	21.000	21.200	21.400	253.800	278.500	280.000
14	Produkt H		14.700	15.000	15.000	14.100	14.200	15.000	14.900	15.400	15.500	15.500	15.700	16.000	181.000	190.400	195.500
15	Gesamt Gruppe II		53.500	54.600	54.800	50.000	50.800	54.400	54.700	55.600	55.300	54.400	55.400	55.700	649.200	700.500	735.200
16																	
17	sonstige Erlöse		7.000	4.500	6.900	7.100	6.800	3.900	4.800	8.100	6.300	7.200	6.900	7.400	76.900	73.200	85.900
18	Umsatzerlöse		215.500	219.100	230.700	234.600	235.100	235.800	237.000	247.700	244.600	244.600	242.400	247.100	2.834.200	3.138.200	3.734.100
19																	
20																	
21	Exportanteil		15,00%	15,00%	15,00%	15,00%	15,00%	15,00%	15,00%	15,00%	20,00%	20,00%	20,00%	20,00%			
22	Umsatzerlöse aus Export		32.325	32.865	34.605	35.190	35.265	35.370	35.550	37.155	48.920	48.920	48.480	49.420	474.065	627.640	749.725
23																	
24	Inladsumsätze Netto		183.175	186.235	196.095	199.410	199.835	200.430	201.450	210.545	195.680	195.680	193.920	197.680	2.360.135	2.510.560	2.984.375
25	Umsatzsteuer	19,00%	34.803,25	35.384,65	37.258,05	37.887,90	37.968,65	38.081,70	38.275,50	40.003,55	37.179,20	37.179,20	36.844,80	37.559,20	448.425,65	477.006,40	567.031,25
26																	
27	Inlandsumsätze Brutto		217.978,25	221.619,65	233.353,05	237.297,90	237.803,65	238.511,70	239.725,50	250.548,55	232.859,20	232.859,20	230.764,80	235.239,20	2.808.560,65	2.987.566,40	3.551.406,25
28																	
29	Umsätze gesamt		250.303,25	254.484,65	267.958,05	272.487,90	273.068,65	273.881,70	275.275,50	287.703,55	281.779,20	281.779,20	279.244,80	284.659,20	3.282.625,65	3.615.206,40	4.301.131,25

Abb.8 Spreadsheet-Design nach Read u. Batson

Bei großen Datenmengen reicht ein einzelnes Arbeitsblatt unter Umständen nicht aus. Würden in einem solchen Fall Daten, Berechnungen und Ergebnisse auf einem einzelnen Arbeitsblatt untergebracht werden, so würde dies zum einen Übersicht und Nachvollziehbarkeit einschränken und zum anderen würden Modifikationen und Aktualisierungen erschwert werden. In diesem Fall ist eine Verteilung auf mehrere Arbeitsblätter sinnvoll. Dabei sollte nach Möglichkeit ein blattübergreifender Aufbau beibehalten werden, bei dem Bezeichner, Variablen, Zeitperioden etc. auf den Blättern immer in den gleichen Spalten zu finden sind.

Raffensperger kritisiert an dieser Vorgehensweise, dass eine modulare Struktur das Spreadsheet unnötig vergrößert und dadurch Zusammenhänge schwerer erkennbar macht. Er empfiehlt, analog zur Kapselung von Daten und ihren Methoden in der Objektorientierung, Daten und Berechnungen zusammenzuhalten [20]. Der Autor zielt damit auf eine Reduzierung der Komplexität und ein besseres Verständnis der Anwendung ab. Das folgende Beispiel einer Personalkostenplanung zeigt eine Umsetzung der Empfehlungen von Raffensperger (vgl. Abbildung 9). Die monatlichen Personalkosten werden für einzelne Unternehmensbereiche unterschiedlich berechnet. Eine räumliche Trennung von Daten, Berechnungen und Ergebnissen würde zu einem unübersichtlichen Spreadsheet führen.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Personalkostenplanung		Jan 08	Feb 08	Mrz 08	Apr 08	Mai 08	Jun 08	Jul 08	Aug 08	Sep 08	Okt 08	Nov 08	Dez 08	2008
2															
3	Produktionsbereich														
4															
5	monatl. Gehalt/Lohn		2.700	2.700	2.700	2.700	2.700	2.700	2.700	2.700	2.700	2.700	2.700	2.700	32.400
6	Anzahl Arbeitnehmer		10	10	10	10	10	10	12	12	14	14	14	14	
7	Monatsbrutto		27.000	27.000	27.000	27.000	27.000	27.000	32.400	32.400	37.800	37.800	37.800	37.800	378.000
8	Urlaubsgeld	500						5000							5.000
9	Weihnachtsgeld	1.500												15000	15.000
10															
11	Rentenversicherung	19,00%	5.130	5.130	5.130	5.130	5.130	5.130	6.156	6.156	7.182	7.182	7.182	7.182	71.820
12	Krankenversicherung	14,00%	3.780	3.780	3.780	3.780	3.780	3.780	4.536	4.536	5.292	5.292	5.292	5.292	52.920
13	Arbeitslosenversicherung	3,00%	810	810	810	810	810	810	972	972	1.134	1.134	1.134	1.134	11.340
14	Pflegeversicherung	2,00%	540	540	540	540	540	540	648	648	756	756	756	756	7.560
15	Sozialversicherungsbeiträge		10.260	10.260	10.260	10.260	10.260	10.260	12.312	12.312	14.364	14.364	14.364	14.364	143.640
16															
17	Kosten Produktionsbereich		37.260	37.260	37.260	37.260	37.260	37.260	44.712	44.712	52.164	52.164	52.164	52.164	521.640
18															
19	Vertriebsbereich														
20															
21	monatl. Gehalt/Lohn		5.300	5.300	5.300	5.300	5.300	5.300	5.300	5.300	5.300	5.300	5.300	5.300	63.600
22	Anzahl Arbeitnehmer		4	4	4	4	4	4	4	4	4	4	4	4	
23	Monatsbrutto		21.200	21.200	21.200	21.200	21.200	21.200	21.200	21.200	21.200	21.200	21.200	21.200	254.400
24	Weihnachtsgeld	4.000												16000	16.000
25															
26	Rentenversicherung	19,00%	4.028	4.028	4.028	4.028	4.028	4.028	4.028	4.028	4.028	4.028	4.028	4.028	48.336
27	Krankenversicherung	14,00%	2.968	2.968	2.968	2.968	2.968	2.968	2.968	2.968	2.968	2.968	2.968	2.968	35.616
28	Arbeitslosenversicherung	3,00%	636	636	636	636	636	636	636	636	636	636	636	636	7.632
29	Pflegeversicherung	2,00%	424	424	424	424	424	424	424	424	424	424	424	424	5.088
30	Sozialversicherungsbeiträge		8.056	8.056	8.056	8.056	8.056	8.056	8.056	8.056	8.056	8.056	8.056	8.056	96.672
31															
32	Kosten Vertiebsbereich		29.256	29.256	29.256	29.256	29.256	29.256	29.256	29.256	29.256	29.256	29.256	29.256	351.072
33															
34	Verwaltungsbereich														
47	Kosten Verwaltungsbereich		9.660	9.660	9.660	9.660	9.660	9.660	9.660	9.660	9.660	9.660	9.660	9.660	115.920
48															
49	Gesamte Personalkosten		76.176	76.176	76.176	76.176	76.176	76.176	83.628	83.628	91.080	91.080	91.080	91.080	988.632
50															
51															

Personalkostenplanung

Abb.9 Spreadsheet-Design nach Raffensperger

Der Ansatz von Raffensperger scheint besonders dann geeignet zu sein, wenn geringe Datenmengen vorliegen, diese jedoch durch eine größere Menge an Formeln verknüpft werden.

Die Beispiele zeigen, dass die Umsetzung bzw. Realisierbarkeit der beiden Ansätze stark von der Aufgabenstellung abhängig ist. So ist auf der einen Seite die Vorgehensweise nach Raffensperger bei datenintensiven Anwendungen weder empfehlenswert noch praktikabel. Andererseits kann sich auch die modulare Einteilung nachteilig auswirken. So besteht die Gefahr, dass Sachverhalte auseinandergerissen und dementsprechend Aufbau und Funktionsweise eines Modells schwerer nachvollziehbar werden. Ein weiterer Nachteil der Modularisierung ist die Notwendigkeit der Konsolidierung der Daten über mehrere Arbeitsblätter oder gar Arbeitsmappen. Das Fehlerrisiko erhöht sich in diesem Fall ebenfalls, da komplexere Referenzen in den Formeln benötigt werden. Wie wir in Abschnitt 2.3 ausgeführt haben, gehören Referenzierungsfehler zur zweit-häufigsten Fehlerkategorie.

Da die allzu starre Anwendung der einzelnen Ansätze je nach Sachverhalt kontraproduktiv sein kann, bietet sich an, diese als Grundstrukturen zu betrachten und ggf. zu kombinieren. *Formelintensive* Spreadsheets legen eine Struktur nach den Empfehlungen von Raffensperger nahe, während *datenintensive* Spreadsheets eher für eine Struktur nach Read und Batson sprechen. Bei Problemstellungen, bei denen sowohl große Datenmengen vorliegen als auch komplexe Berechnungen notwendig sind, könnte eine Mischform in Betracht gezogen werden, bei der bspw. mehrere Arbeitsblätter oder

mehrere Arbeitsmappen genutzt werden, innerhalb derer eine Modellierung nach den Empfehlungen von Raffensperger verfolgt wird.

Neben den Überlegungen zum Aufbau des Spreadsheets ist die Implementierung von Formeln ein weiterer wesentlicher Punkt im Rahmen der Maßnahmen zur Fehlerreduzierung. Die Grenzen zwischen Spreadsheet-Design und der Implementierung der Formeln sind dabei fließend. Ein Design, welches pro Zeile und Spalte nur den Einsatz einer einzigen Formel vorsieht, erleichtert Kopiervorgänge und reduziert Fehler, da bedenkenlos über den gesamten Bereich kopiert werden kann ohne dabei versehentlich Formeln zu überschreiben. Formeln sollten dabei nur auf Zellen links und oberhalb der eigenen Position referenzieren. Eine solche Anordnung entspricht dem Lesefluss und fördert das Verständnis für die Anwendung [20].

Neben der Positionierung der Formeln ist auch ihre Lesbarkeit und Wartbarkeit zu berücksichtigen. Read und Batson empfehlen, komplexe Formeln für eine bessere Lesbarkeit über mehrere Zeilen anzuordnen. Diese Vorgehensweise wird von Raffensperger kritisiert. Er vertritt die Meinung, dass auch bei der Implementierung der Formeln das Lokalitätsprinzip gelten sollte, weil eine Aufspaltung von Formeln das Verständnis erschwere. Nach unserer Auffassung ist auch hier eine Mischform angezeigt. Eine sehr lange Formel wird an Lesbarkeit gewinnen, wenn sie über mehrere Zellen verteilt wird – allerdings sollte die Zerlegung nur dann erfolgen, wenn in jeder Zelle ein sinnvolles Teilergebnis berechnet werden kann, welches für sich verständlich ist. Bei der Implementierung von Formeln ist der Einsatz von Bereichsnamen empfehlenswert. Bei sinnvoll gewählten Namen werden Zusammenhänge besser verständlich, und die Lesbarkeit wird erhöht. Durch die Namensvergabe für einzelne Zellen oder Gruppen von Zellen wird die Arbeit mit Referenzen erheblich vereinfacht. Gerade bei Referenzen zwischen Arbeitsblättern oder Arbeitsmappen wird die Fehleranfälligkeit reduziert (z. B. `ZINSSATZ` anstelle von `Tabelle1!$A$1`). Aktualisierungen sind durch die Namensverwaltung leichter durchführbar. Sollen Formeln auf eine neue Zelle verweisen, kann einfach die Definition des Namens geändert werden. Bei der Verwendung von mehreren Blättern ist darüber hinaus die Definition von Namensräumen sinnvoll (`BLATT_X.WERT_Y`).

Der Einsatz von Konstanten in Formeln (Hard-coding) ist zu vermeiden, da diese bei Aktualisierungen leicht übersehen werden können (Rajalingham et al. ordnen solche Fehler der Maintainbility-Klasse zu). Wie aus Abschnitt 2.3 deutlich wird, hat dieses Defizit aber nicht nur in der Phase der Verwendung und Modifikation negative Auswirkungen, sondern bereits in der Entwicklungsphase: Hard-coding ist die häufigste Fehlerkategorie. Um Hard-coding zu vermeiden, empfehlen Read und Batson eine eigene Spalte für Konstanten anzulegen und auf diese zu referenzieren. Bei Aktualisierungen müssen dann lediglich die Werte in diesem Bereich angepasst werden. In Abbildung 9 ist diese Vorgehensweise beispielhaft zu sehen. Anstatt die Anteile für die Sozialversicherungsbeiträge direkt in die entsprechenden Formeln zu implementieren, werden sie in Spalte B gesondert aufgeführt.

Die in diesem Abschnitt umrissenen Maßnahmen greifen weitestgehend auf Funktionalitäten zurück, die in der gängigen Tabellenkalkulationssoftware vorhanden sind. Der Umgang mit Skriptsprachen, wie z. B. VBA, wird an dieser Stelle nicht behandelt. Dies würde den Rahmen dieses Beitrages sprengen. Es steht jedoch außer Frage, dass dem Einsatz von Skriptsprachen, insbesondere bei umfangreichen Anwendungen, große Bedeutung zuteil wird.

3.3 Kontrolle

Auch wenn sich viele Fehlerquellen bereits durch die vorgelagerten Phasen unterbinden lassen, ist die Entstehung von Fehlern dadurch selbstverständlich nicht ausgeschlossen. Daher ist eine Kontrolle des Spreadsheets nach Fertigstellung (bzw. auch während der Entwicklung) oder Modifikation notwendig. Die von Powell et al. durchgeführte Studie zeigt, dass der Großteil der Unternehmen dabei nicht einem zuvor ausgearbeiteten Testplan folgt, sondern nach persönlichem Empfinden (gesunder Menschenverstand) vorgeht [18]. Sporadisches Testen führt jedoch selten zum gewünschten Erfolg. Eine durchdachte Testphase ist daher empfehlenswert. Zur Strukturierung der Testphase empfehlen Read und Batson [16] die Erstellung eines Testplans, in dem die einzelnen Schritte (Maßnahmen) der Testphase beschrieben werden. Im Idealfall werden die Tests dabei von Personen durchgeführt, die nicht an der Entwicklung des Spreadsheets beteiligt waren. Dabei sollte schon während der Entwicklung getestet werden, ob die Anforderungen der Spezifikation erfüllt sind, um Fehler im Modell auszuschließen. Für die Testphase kommen sowohl softwaregestützte Verfahren in Form von Auditing-Werkzeugen als auch Maßnahmen, die keiner zusätzlichen Software bedürfen, in Betracht. So gibt es z. B. Werkzeuge, die es ermöglichen, in Formelansicht des Spreadsheets zwischen Originalformeln und Kopien zu unterscheiden. Darüber hinaus ist bspw. MS Excel von Hause aus bereits mit einer ganzen Reihe von Auditing-Funktionen ausgestattet (z. B. Anzeigen von Zirkelverweisen, Pfeile zu Vorgängern/Nachfolgern etc.).

Als „manuelle“ Maßnahmen empfehlen Read und Batson die Überprüfung von Formeln durch Nachrechnen. Zusammenhänge können durch die Formelansicht und das Anzeigen von Abhängigkeiten zwischen den Zellen verdeutlicht werden. Eine weitere Maßnahme sind Belastungstests, bei denen extreme Eingabewerte oder extreme berechnete Werte überprüft werden. Ebenso sollte sichergestellt werden, dass falsche Eingaben keine korrekt erscheinenden Ergebnisse liefern. Bei der Verwendung von Skriptsprachen ist deren korrekte Funktionsweise in verschiedenen Situationen ebenfalls zu prüfen. Im Rahmen der Testphase muss neben der mathematischen Korrektheit der Ergebnisse auch die korrekte Implementierung des Modells zu überprüft werden. Dies kann bspw. durch einen Vergleich der Ergebnisse mit denen aus Vorgängermodellen geschehen.

Ergänzen lassen sich diese Maßnahmen durch den Einsatz von Auditing-Werkzeugen. Solche Werkzeuge erleichtern das systematische Testen des Spreadsheets. Es gibt

vielfältige Ausprägungen solcher Werkzeuge[7]. An dieser Stelle soll ein softwaregestützter Testvorgang exemplarisch an dem Excel Add-In Audit vorgestellt werden[8]. Die Funktionsweise von Audit ist nur eine von vielen Möglichkeiten für softwaregestützte Testvorgänge[9].

Die folgende Abbildung 10 zeigt die Anwendung von Audit an dem in Kapitel 3 vorgestellten Beispiel nach Raffensperger. Um den Einsatz des Werkzeuges besser zur Geltung zur bringen, wird weitestgehend auf die farbliche Hervorhebungen von Zellen verzichtet. Dadurch wird gleichzeitig sichtbar, in welchem Maße die farbliche Hervorhebung das Verständnis für den Aufbau des Spreadsheets verbessert hat.

Abb.10 Anwendung von Audit

Logisch zusammenhängende Zellen werden von dem Werkzeug zu Regionen zusammengefasst (gekennzeichnet durch lilafarbende Umrandungen). Diese Regionen werden vom Tester auf Fehler überprüft. Der Wert einer Zelle kann dabei vom Tester als richtig oder falsch markiert werden. Jeder dieser Zustände führt dazu, dass Zellen, die Einfluss auf den Wert der markierten Zelle haben, farblich hervorgehoben werden. Im oberen Bereich des Spreadsheets ist zu erkennen, dass die Werte in C7 und O5 vom Tester als korrekt markiert wurden. Als Folge dessen werden alle Werte, die zu dem Ergebnis in den beiden Zellen (abgeleitete Werte) führen, blau unterlegt. Im Vertriebsbereich liegt jedoch ein Fehler vor. Der Wert in der Zelle D27 wurde

---

[7] Siehe http://www.sysmod.com/sslinks.htm#auditing [Abruf am 9.11.2009] für eine umfangreiche Auflistung.

[8] http://www.redroversoftware.com/products/audit/ [Abruf am 9.11.2009]

[9] Audit basiert auf dem *What-You-See-Is-What-You-Test*-Verfahren. Eine Vorstellung dieses und weiterer Verfahren findet sich in [21].

fälschlicherweise durch die Formel der darüber gelegenen Zelle berechnet. Dem Tester fällt dieses durch eine Abweichung in der Struktur auf, da D27 von dem Werkzeug als eigene Region gekennzeichnet wird. Indem der Wert der Zelle und folgerichtig das Endergebnis in D32 als falsch gekennzeichnet werden, zeigen sich mögliche Kandidaten für die Fehlerentstehung. Die Zelle D27 wird als wahrscheinlichste Fehlerquelle identifiziert (Rotfärbung). Durch die Validation von weiteren Zellen können die Wahrscheinlichkeiten von der Software noch genauer bestimmt werden.

Der Einsatz von Auditing-Werkzeugen ermöglicht ein relativ schnelles Testen der Anwendung, sodass auch bei knapper Entwicklungszeit nicht auf den softwaregestützten Test verzichtet werden sollte. Allerdings bieten Auditing-Werkzeuge i. d. R. lediglich die Möglichkeit, auf mathematische Fehler oder schlechte Praktiken wie die Verwendung von Konstanten aufmerksam zu machen. Die Korrektheit des Modells lässt sich dagegen nicht überprüfen, sodass eine Kombination mit den „manuellen“ Verfahren sinnvoll ist. Es sei angemerkt, dass sowohl durch Werkzeug-unterstütztes als auch durch „manuelles“ Testen lediglich die Anwesenheit von Fehlern nachgewiesen werden kann und damit eine Fehlerbeseitigung möglich wird. Der Nachweis der Abwesenheit von Fehlern ist mit Tests nicht möglich.

3.4 Verwendung und Modifikation

Die in Kapitel 2 vorgestellte Klassifikation von Rajalingham et al. berücksichtigt mit der Rolle des Datenempfängers die Fehlerentstehung außerhalb der Entwicklungsphase. In den Phasen der Verwendung und der Modifikation sind sowohl die Dokumentation als auch die Ergreifung von Maßnahmen zum Schutz gegen unabsichtliche Veränderungen als besonders wichtige Punkte hervorzuheben. Neben der Angabe von Informationen zu den Entwicklern (Name, Abteilung etc.) sollte in der Dokumentation auch die Versionsnummer festgehalten werden (Read und Batson empfehlen die Dokumentation auf einem separaten Blatt zu platzieren). Diese Angaben sollten als Mindestumfang betrachtet werden und auch bei knapper Entwicklungszeit erfolgen. Der Zweck der Anwendung und eine Anleitung zur Verwendung anhand eines Beispiels sollten ebenfalls eingefügt werden. Darüber hinaus sollte eine technische Dokumentation mit Designdetails inklusive Changelog und Hinweisen zur Erweiterung erfolgen. Makros sollten ebenfalls dokumentiert werden. Neben der Erstellung eines Dokumentationsblattes kann auch die Kommentarfunktion der Tabellenkalkulationssoftware genutzt werden, um einzelne Eingaben, Ausgaben oder Formeln zu dokumentieren. Für die Formeldokumentation kann alternativ eine eigene Spalte angelegt werden. In Hinblick auf die Lebensdauer und Wiederverwendbarkeit und auch auf die Kosten der Erstellung erscheint eine gute Dokumentation unerlässlich. In zeitkritischen Situationen sollte darüber nachgedacht werden, nachträglich eine umfangreichere Dokumentation zu erstellen.

Zur Verhinderung unzulässiger Änderungen kann der Zellschutz aktiviert werden. Die Kenntlichmachung von Eingabewerten kann einer fehlerhaften Benutzung des Spreadsheets vorbeugen. Sensible Daten können durch die Einrichtung eines Passworts geschützt werden. Da diese Maßnahmen ohne großen Zeitaufwand durchführbar sind, sollten sie als obligatorisch betrachtet werden.

4 Schlussbemerkung

Auch wenn für das Spreadsheet-Engineering eine ganze Reihe von Parallelen zum Software-Engineering existieren, muss beachtet werden, dass sich mit der Tabellenkalkulation auch ohne besondere Vorkenntnisse Ergebnisse produzieren und vor allem auch modifizieren lassen. Auf der einen Seite ist es notwendig, den Benutzer der Tabellenkalkulation weiter für die Fehlerentstehung zu sensibilisieren und Wege zur Fehlervermeidung aufzuzeigen. Auf der anderen Seite dürfen die Empfehlungen zur Fehlervermeidung nicht so restriktiv oder aufwendig in der Umsetzung sein, dass der flexible Charakter der Tabellenkalkulation verloren geht oder gar eine Ausbildung zum professionellen Programmierer vorausgesetzt werden muss.

5 Literaturverzeichnis

[1] Seufert, Andreas and Martin, Wolfgang. Unternehmenssteuerung und Business Intelligence im Mittelstand 2008. [Online] 2009. [Cited: 11 12, 2009.] http://www.erpmanager.de/magazin/artikel_1988-print_studie_business_intelligence.html.

[2] Eckerson, Wayne W. Beyond Reporting: Delivering Insights with Next-Generation Analytics. [Online] 2009. [Cited: 11 12, 2009.] http://www.tdwi.org/research/reportseries/reports.aspx?pid=774.

[3] Reuters. TransAlta Says Clerical Snafu Costs It $24 Million. The Globe and Mail. [Online] 2003. [Cited: 11 12, 2009.] http://www.globeinvestor.com/servlet/ArticleNews/story/ROC/20030603/2003-06-03T232028Z_01_N03354432_RTRIDST_0_BUSINESS-ENERGY-TRANSALTA-COL.

[4] Rose, Jason. Taking Human Error Out of Financial Spreadsheets. Strategic Finance. 2007, Vol. 88, 9, pp. 52-55.

[5] Burnett, Margaret; Cook, Curtis and Rothermel, Gregg. End-User Software Engineering. Communications of the ACM. 2004, Vol. 47, 9, pp. 53-58.

[6] Bals, Jan-Christopher; Christ, Fabian; Engels, Gregor; Sauer, Stefan. Softwarequalität überall - Excel-lente Software. Forschungsforum Paderborn 10. 2007, 1, S. 50-60.

[7] Spreadsheet Engineering: A Research Framework. Grossman, Thomas A. Cardiff, Wales, 2002. Proceedings of the European Spreadsheet Risks Interest Group Symposium. pp. 23-34.

[8] Powell, Stephen G.; Baker, Kenneth R. and Lawson, Barry. A critical review of the literature on spreadsheet errors. Decision Support Systems. 2008, Vol. 46, 1, pp. 128-138.

[9] Panko, Raymond R. and Halverson, Richard P. Jr. Spreadsheets on trial: a survey of research on spreadsheet risks. Hawaii, 1996. Proceedings of the 29th Annual Hawaii International Conference on System Sciences. pp. 326-335.

[10] Panko, Raymond R. Revisiting the Panko-Halverson Taxonomy of Spreadsheet Errors. London, 2008. Procedings of the European Spreadsheet Risks Interest Group, EuSpRIG 2008 Conference. pp. 199-220.

[11] Panko, Ramond R. What we know about spreadsheet errors. Journal of End User Computing. 1998, Vol. 10, 2, pp. 15-21. Der Beitrag wird vom Autor unregelmäßig auf seiner Website unter http://panko.shidler.hawaii.edu aktualisiert.

[12] Rajalingham, Kamalasen; Chadwick, David R. and Knight, Brian. Classification of spreadsheet errors. Grennwich, England, 2000. Proceedings of the European Spreadsheet Risks Interest Group. pp. 23-34.

[13] Powell, Stephen G.; Baker, Kenneth R. and Lawson, Barry. Impact of errors in operational spreadsheets. Decision Support Systems. 2009, Vol. 47, 2, pp. 126-132.

[14] Powell, Stephen G.; Baker, Kenneth R. and Lawson, Barry. An auditing protocol for spreadsheet models. Information & Management. 2008, Vol. 45, 5, pp. 312-320.

[15] Powell, Stephen G.; Baker, Kenneth R. and Lawson, Barry. Errors in Operational Spreadsheets. Journal of Organizational and End User Computing. 2009, Vol. 21, 3, pp. 24-36.

[16] Read, Nick and Batson, Jonathan. Spreadsheet modelling best practice. [ed.] Institute of Chartered Accountants in England & Wales. London, 1999.

[17] Madahar, Mukul; Cleary, Pat and Ball, David. Categorisation of Spreadsheet Use within Organisations, Incorporating Risk: A Progress Report. University of Greenwich, London, 2007. Proceedings of the European Spreadsheet Risks Interest Group, EuSpRIG 2007 Conference. pp. 37-45.

[18] Baker, Kenneth R.; Foster-Johnson, Lynn; Lawson, Barry; Powel, Stephen G.. Spreadsheets Risk, Awareness and Control. SPREADSHEET ENGINEERING RESEARCH PROJECT. [Online] [Cited: 11 13, 2009.] http://mba.tuck.dartmouth.edu/spreadsheet/product_pubs.html/SSRisk.doc.

[19] Bewig, Philip L. How do you know your spreadsheet is right?: Principles, Techniques and Practice of Spreadsheet Style. European Spreadsheet Risks Interest Group. [Online] 2005. [Cited: 8 27, 2009.] http://www.eusprig.org/hdykysir.pdf.

[20] Raffensperger, John F. New Guidelines for Spreadsheets. International Journal of Business and Economics. 2003, Vol. 2, 2, pp. 141-154.

[21] Jacobs, Jürgen und Tiburtius, Phillip. Testmethoden zur Qualitätssicherung von Tabellenkalkulationsanwendungen. Facettenreiche Wirtschaftsinformatik: Controlling, Compiler & more, Festschrift für Prof. Dr. Karl Goede, FINAL. 2009, Bd. 19, 1, S. 35 – 49.

Reflexionen über eine strategiegetriebene, adaptive Budgetierung

Prof. Dr. Sven Piechota

1 Grundlagen

1.1 Einordnung von Strategie und Budgetierung in die Unternehmensplanung

Budgetierung und Strategie sind im Kontext der Unternehmensplanung zu sehen. Unternehmensplanung ist eine zentrale Aufgabe der Unternehmensführung. Durch Planung sollen Entscheidungen, die einen zukünftigen Zeitraum betreffen und daher von Unsicherheit geprägt sind, optimiert werden, und die Organisation soll laufend an interne und externe Veränderungen angepasst werden. Zu verstehen ist unter Planung daher ein „...systematisches, zukunftsbezogenes Durchdenken von Zielen, Maßnahmen, Mitteln und Wegen zur zukünftigen Zielerreichung“[1]. Planung kann dabei durch die Adjektive zukunftsbezogen, gestalterisch, rational, informationell, prozessual, zielorientiert und interdependent charakterisiert werden.[2]



Abb. 1: Strategische Führung als analytischer Managementprozess

[1] *Wild, J.*, Grundlagen der Unternehmensplanung (B), 1982, S. 13, zitiert nach *Buchner, H.*, Planung im turbulenten Umfeld (B), 2002, S. 11.
[2] *Buchner, H.*, Planung im turbulenten Umfeld (B), 2002, S. 11 f.

Aus dem Qualitätsmanagement kennen wir den „Plan-Do-Check-Act“-Zyklus als grundlegendes Handlungsmuster von Führung. In diesem Sinn kann die strategische Führung als stetige Abfolge von Führungstätigkeiten des

- Ausrichtens („Der strategische Plan“)
- Agierens
- Analysierens
- Anpassens

verstanden werden. Dementsprechend versteht man unter der strategischen Führung den analytischen Führungsprozess, der die Teile

- Strategiefindung
- Strategieexekution
- Strategische Kontrolle

umfasst.

Unter **strategischer Führung** ist idealisierend ein rationaler Prozess zu verstehen, der die Formulierung, Implementierung und Kontrolle von Strategien zur Aufgabe hat. Zum besseren Verständnis der strategischen Führung wird unten der Begriff „Strategie“ ausführlich erläutert. Die **operative Führung** beschäftigt sich hingegen mit der Entwicklung und Umsetzung der notwendigen Maßnahmen.

Zusätzlich soll die Unterteilung der Planung in Sach- und Formalzielorientierung erwähnt werden. Die **sachzielorientierte** Planung geht von qualitativen Zielsetzungen aus (Planung von Aktionen und Maßnahmen), wohingegen die **formalzielorientierte** Planung die Aktionen und Maßnahmen quantitativ darstellt. Die **formalzielorientierte,** in wertmäßigen Größen ausgedrückte Planung ist die Budgetierung[3], mit der sich dieses Kapitel weiter unten näher beschäftigt. Die Budgetierung wird dabei als primäres Steuerungsinstrument einer Unternehmung aufgefasst, das grundsätzlich das Ende der kurzfristigen operativen Planung darstellt und die Leistungskontrolle beinhaltet.
Die **sachzielorientierte** operative Planung soll als Verbindung zwischen strategischer Planung und Budgetierung die Operationalisierungslücke zwischen der strategischen Planung und der Budgetierung schliessen: Strategien sind in operative Maßnahmen umzusetzen, die die Grundlage für Budgets sind. Die Überbrückung dieser Lücke ist die Voraussetzung einer nachhaltig erfolgreichen Unternehmensführung.

[3] *Horváth, P.*, Controlling (B), 2003, S. 231.

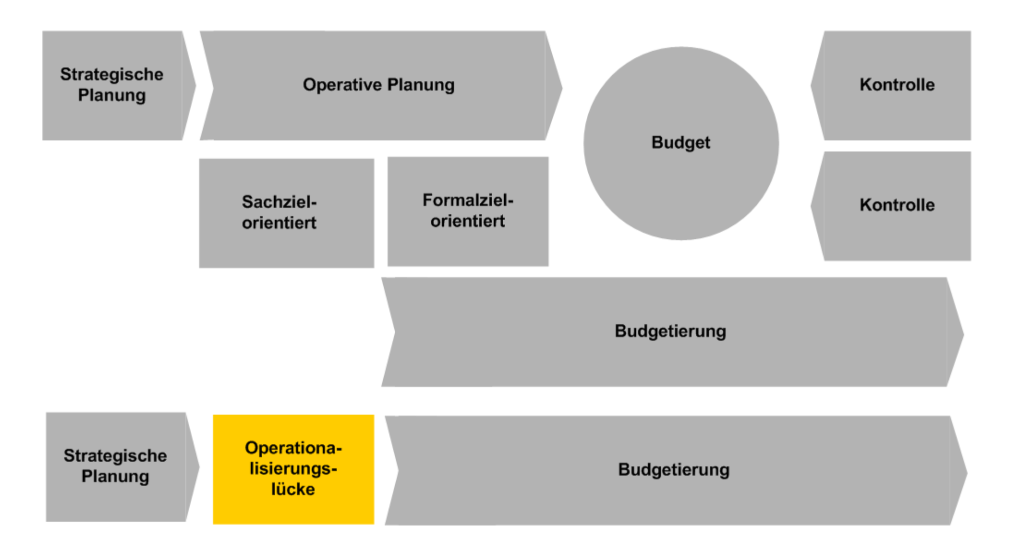


Abb. 2: Operationalisierungslücke der Budgetierung

1.2 Was sind Strategien und warum sind sie nützlich?

Was sind Strategien?

Strategie ist ein weitläufiges, beinahe universelles Konzept. Nicht nur Unternehmen haben Strategien (oder sollten sie haben), sie finden sich auf allen Ebenen des gesellschaftlichen Lebens, von der individuellen Lebens- oder Karriereplanung bis hin zu der grundlegenden staatlichen Überlegung, wie Wohlstand auf volkswirtschaftlicher Ebene entstehen und verwendet werden soll. Strategen wollen mit Strategien Zukunft gestalten, Strategien sind mit dem Anspruch verbunden, eine potenziell bessere Zukunft erreichen zu wollen. Deshalb haben Strategien ebenso viel mit Entschlossenheit, Wille und Glauben zu tun wie mit Analyse, Planung und Umsetzung. Das Management von Strategien hat folglich eine „harte“ Seite, in der es um konkrete Fragen der Strategiefindung und –umsetzung wie Ziele, Zielhöhen und Zielerreichung geht, aber auch eine „weiche“ Seite, in der Führung, Zusammenarbeit, Motivation und Unternehmenskultur eine wichtige Rolle spielen.

Für Unternehmen ist die Strategieentwicklung und –umsetzung der Kern der unternehmerischen Führung. Eine Unternehmensstrategie besteht aus den Schachzügen, dem Spielplan, mit dem Wachstum erzielt, eine Marktposition erreicht, Kunden gewonnen und gebunden, der Wettbewerb bestanden, das Unternehmen organisiert und die Ziele erreicht werden sollen. Der Zweck einer Unternehmensstrategie ist damit, durch Aktionen eine gewünschte Wettbewerbs-position und finanzielle Leistungsstärke zu erreichen und zu erhalten und - im Idealfall - Wettbewerbsvorteile gegenüber Rivalen zu erlangen, die die Möglichkeit von überdurchschnittlichen Renditen schaffen. Eine Unternehmensstrategie entwickelt und verändert sich im Zeitablauf, sie geht entweder aus proaktivem Verhalten des Managements oder als Reaktion auf unerwartete Entwicklungen im Umfeld oder Markt des Unternehmens hervor.

Die sprachlich-historischen Wurzeln des Strategiebegriffes liegen in der Militärwissenschaft und sind dort auf das griechische Wort „stratégos“

zurückzuführen, was „Heeresführer" bedeutet.[4] Seit der Mitte des 20. Jahrhunderts fand dann auch die Transformation des Begriffes in der Betriebswirtschaftslehre und hier zunächst im Rahmen der Spieltheorie statt. Heute ist die wissenschaftliche Befassung mit Fragen der strategischen Unternehmensführung ein fester Bestandteil der theoretischen Betriebswirtschaftslehre und einer praxisorientierten Managementlehre.

Im weiteren Verlauf soll Strategie in Anlehnung an das klassische Strategieverständnis als ein grundsätzliches, langfristiges Handlungsmuster von Unternehmen und relevanten Teilbereichen gegenüber ihrer Umwelt zur Verwirklichung langfristiger Ziele verstanden werden.

Strategien treffen Aussagen zu folgenden Bereichen:

- **Partizipationsaussage**: Welches Leistungsangebot (Produkte/ Dienstleistungen) will das Unternehmen an welche Zielkunden für welches Kundenproblem in welchem Zielmarkt wertschaffend bereitstellen?
- **Positionierungsaussage**: Welche Markt- und Wettbewerbsposition strebt das Unternehmen an, und wie soll diese operationalisiert werden (Wettbewerbsvorteil)?
- **Das strategische Modell**: Welche ***strategischen Ziele*** (finanziell und nicht-monetär) will das Unternehmen erreichen, welche Zielcluster wirken erfolgskritisch (strategische Erfolgsfaktoren)? Welche (angenommene) ***Wirklogik*** besteht zwischen den strategischen Zielen oder Erfolgsfaktoren?
- Wie soll das **Wertlieferungssystem** des Unternehmens konfiguriert werden, wie soll die Wertschöpfungskette intern gestaltet und extern angebunden werden, wie soll die Ablauf- und Aufbauorganisation gestaltet werden?
- Worin besteht die **Wertbasis** des Unternehmens? Über welche immateriellen und materiellen Ressourcen soll das Unternehmung verfügen und welche Fähigkeiten und Kenntnisse sollen das Unternehmen auszeichnen?

Strategien stellen folglich Weg-Zielbeschreibungen dar, die als Handlungsmuster für alle Entscheidungen und Aktivitäten im Unternehmen richtungsgebend wirken sollen. Sie beschreiben das Ergebnis von Entscheidungen eines Unternehmens, die sich aus den Festlegungen in den verschiedenen Aussagebereichen einer Strategie ergeben. Visionen fungieren für strategische Ziele quasi als Metaziele, an denen sie sich auszurichten haben.

Durch die Verwirklichung der Vision erlangt das Unternehmen eine Position, durch die es sich vom Wettbewerb differenziert.[5] Diese wird durch die Konzentration auf die spezifischen strategischen Ziele erreicht, die für die angestrebte Differenzierung von herausragender Bedeutung sind. Die Vision erhält somit eine konkrete Form und wird in handhabbare Ziele zerlegt, die gemanagt werden können. Die Entwicklung der strategischen Ziele und ihre Erreichung sollten folglich dazu führen, dass sich der gegenwärtige Zustand des unternehmerischen Handelns in den wünschenswerten wandelt. Laut Porter kann ein Unternehmen allerdings nur seine Visionen erreichen,

[4] Bea F.X./Haas J., Grundwissen, 2001, S. 50.; vgl. auch Welge, M./Al-Laham, A., Management, 2001, S. 12.

[5] *Porter, M.*, Wettbewerbsstrategie (B), 1999, S. 21.

wenn es bereit ist, alle Entscheidungen (Trade Offs) und Aktivitäten auf die Schaffung eines einzigartigen, von den Kunden erkennbaren Leistungsangebotes auszurichten.[6]

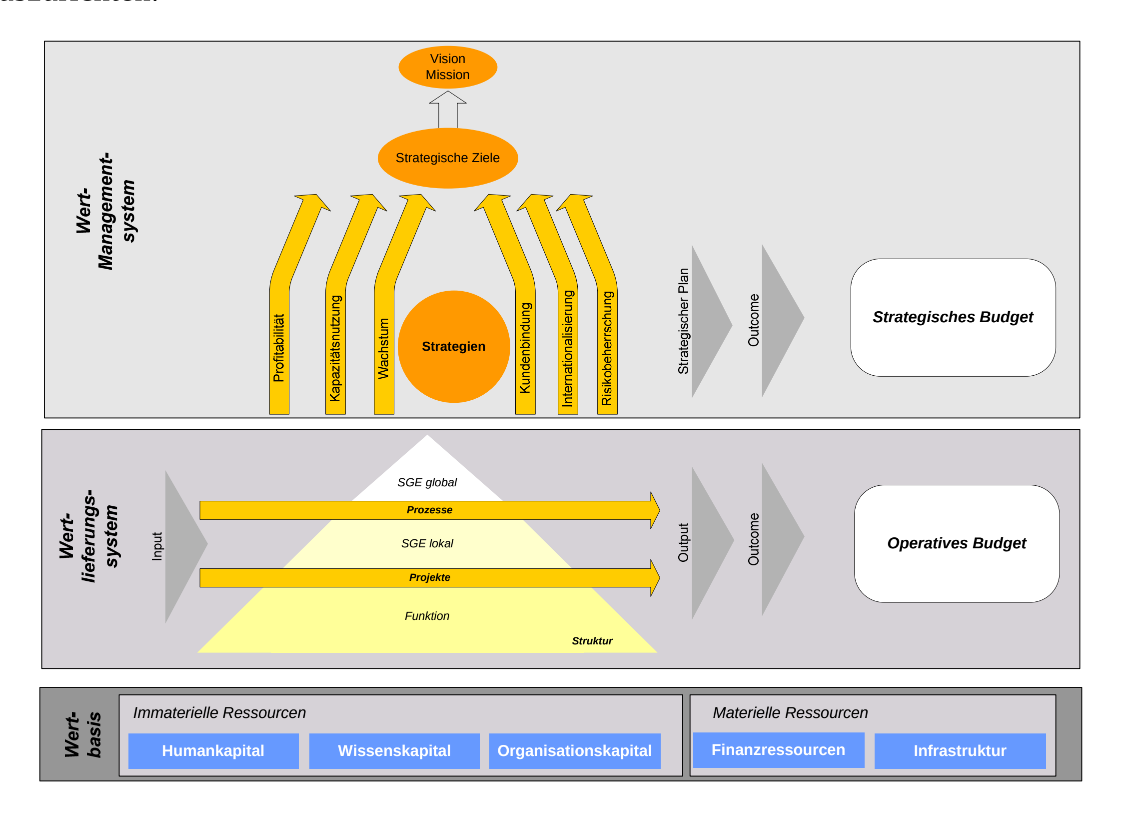


Abb. 3: Strategie

In diesem Lichte kann man strategische Unternehmensführung als die Kunst der nachhaltigen Wertschaffung eines Unternehmens für seine Kunden, Wertschöpfungszulieferer (Lieferanten, Mitarbeiter) und Anteilseigner sehen. Die Basis hierfür stellen die immateriellen und materiellen Ressourcen eines Unternehmens dar. Ressourcen bilden ein Reservoir für wertschaffende Unternehmensleistungen, die in einem Wertlieferungssystem organisiert werden müssen. Hierdurch entstehen hierarchisch gegliederte Aufbau- und Ablaufstrukturen (Abb.3). Das Wertmanagementsystem richtet die organisierten Ressourcen mittels Strategien auf die strategischen Ziele, letztlich auf die in der Vision formulierten nachhaltigen Unternehmensziele aus.

1.3 Strategien in Wandel

Über viele Jahre hat sich die wirtschaftswissenschaftliche Forschung und Lehre damit beschäftigt, welcher Grundausrichtung Strategien angesichts der Käufermärkte in den spätindustriellen Wirtschaften folgen sollten. Das strategische Management wurde grundlegend von Porters Arbeiten über Wertketten und Wettbewerbsstrukturen geprägt: dominantes Metaziel für die Formulierung von Strategien war die Schaffung von ***Wettbewerbsvorteilen***. Industrien und innerbetriebliche Wertschöpfungen können als arbeitsteilige Prozesse verstanden werden, in denen Zulieferer Wert für

[6] *Porter, M.*, What Is Strategy? (A), 1996, S. 70.

Unternehmen schaffen und im Gegenzug hierfür Erlöse realisieren, und Unternehmen mit ihren Leistungsangeboten Werte für Kunden schaffen und Gegenwerte erhalten. In der marktwirtschaftlichen Wirtschaftsform geschieht dies immer im Wettbewerb, entweder im faktischen Wettbewerb zwischen Unternehmen derselben Branche oder im potenziellen Wettbewerb mit Unternehmen benachbarter Branchen oder vor- bzw. nachgelagerter Wertschöpfungsstufen, die bei ausreichender Attraktivität in die Branche eines Unternehmens einzudringen gewillt sind.

Wettbewerbsstrategien zielen in diesem Umfeld grundsätzlich darauf, Vielfalt in den strategischen Optionen einzugrenzen oder zu beschneiden: die Unternehmen selbst legen sich in ihren Strategien langfristig fest, sie wollen hierdurch eine drohende Sprunghaftigkeit in der Ausrichtung ihrer Organisationen vermeiden. Im Umgang mit Wettbewerbern werden Strategien darauf ausgerichtet, strategische Optionen für den Wettbewerber zu verstellen: Markteintrittsbarrieren werden eingerichtet, um den Handlungsraum der Konkurrenten so einzugrenzen. Insgesamt führt die Orientierung an verteidigungsfähigen Wettbewerbsvorteilen in spätindustriellen Wirtschaften zu konservativen, defensiven Strategieausrichtungen, die langfristig bestehen bleiben sollen und nur bei Veränderungen im Wettbewerbsumfeld angepasst werden.

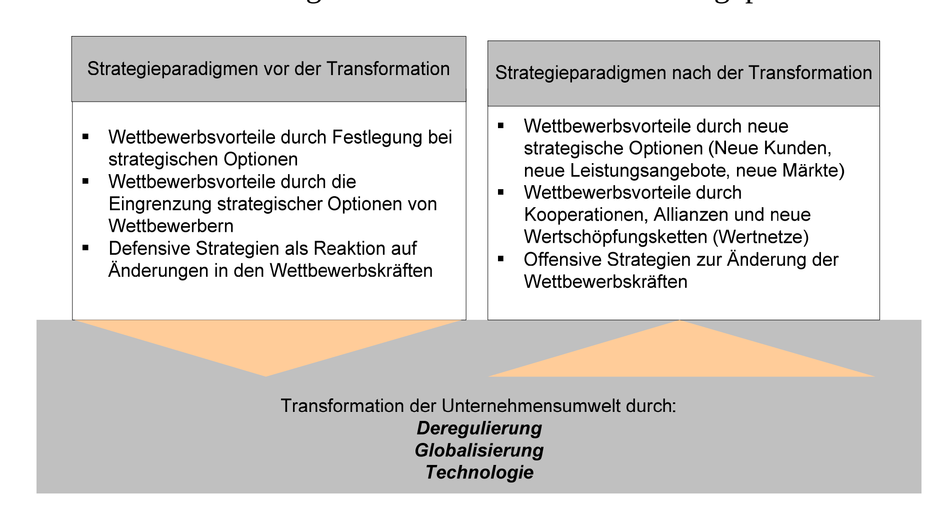


Abb. 4: Strategien im Wandel

Ende der 20. Jahrhunderts treten in den relevanten Weltökonomien wesentliche Änderungen des Wettbewerbsumfeldes ein: auf nationaler und internationaler Ebene führen **Deregulierungen** zu einer steigenden Internationalisierung des wirtschaftlichen Geschehens und des Arbeitsmarktes, das Wettbewerbsrecht wird liberalisiert, Zölle fallen weg und staatlich betriebene Monopole werden aufgelöst. **Globalisierung** wird dieses Phänomen genannt, bei dem Unternehmen die Welt als Beschaffungs- und Absatzmarkt definieren, international homogene Moden entstehen und homogene Kundenpräferenzen auszumachen sind. Die Internationalisierung der Kapitalmärkte schreitet voran und führt zu einem stark erhöhten Leistungsdruck auf allen Ebenen in den Unternehmen und zu einer Internationalisierung von Managementmethoden. **Technologieschübe** in der Informationsverarbeitung führen zu freien Informationszugängen und zu neuen Kommunikations- und Kollaborationsformen, eBusiness entsteht, bei dem Unternehmen die Wertschöpfung untereinander oder die Zusammenarbeit mit den Kunden auf neue Medien umstellen können. Oftmals werden tradierte physikalische Verkaufs- und Servicestellen zur

wirtschaftlichen Belastung. Neue Märkte entstehen, langsamer als in der ersten Euphorie des Internetbooms erwünscht, aber nachhaltig. Neue Werkstoffe und Transporttechniken bewirken neue Leistungsangebote und effiziente Arbeitsformen in der Logistik.

In diesem Umfeld erfolgt ein deutlich bemerkbares Überdenken der auf die defensive, wenig bewegliche und einseitig auf Wettbewerbsvorteile fokussierten Formulierung von Strategien. Strategische Flexibilität ist gefordert, um die Wahrnehmung und Reaktionsfähigkeit der Unternehmen auf die Veränderungen in den Industrien und Märkten zu sichern. Die Beobachtungszyklen des Unternehmensumfeldes werden von einer jährlichen Ausprägung auf eine ständige Überwachung des externen Geschehens umgestellt, die Diskussion und Bewertung strategierelevanter Sachverhalte wird offen und mit einer externen Perspektive geführt, strategische Entscheidungen und Schwerpunkte werden wie ein Portfolio von Optionen angesehen und permanent neu priorisiert, die Ergebnisse von Strategien permanent beobachtet und das gesamte Wertmanagementsystem als lernendes System ausgestaltet und letztlich: die Ressourcenzuteilung aufgrund von Strategiewechsel wird beschleunigt. Das gewandelte Wirtschaftsumfeld verstärkt diese Tendenz der Komplexitätssteigerung und Flexibilisierung von Strategien: Neue Zielmärkte, neue Zielkunden und neue Leistungsangebote bewirken, dass die Zahl der strategischen Optionen erhöht wird und deshalb einen erhöhten Steuerungsbedarf im strategischen Management zur Folge hat: Innovationen sind gefragt und Wettbewerbsvorteile hat nicht das Unternehmen, das Innovationsmöglichkeiten verhindert, sondern sie entdeckt und gestaltet. Andere Wettbewerbsvorteile sind durch Allianzen, Kooperationen oder neue Wertschöpfungsgestaltungen bzw. –gemeinschaften möglich. All dies führt zu sehr viel offensiver zu formulierenden Strategien und zu einer „Strategisierung“ der Unternehmensführung, in der Formulierung, Bewertung, Kommunikation, Implementierung und Erfolgskontrolle von Unternehmensstrategien eine erhöhte Bedeutung für die Überlebensfähigkeit von Unternehmen haben.

2 Budgetierung

Strategien werden in das Planungssystem von Unternehmen integriert. Sie stellen den qualitativen Teil eines Planungssystems dar, in dem die Entscheidungen für die jeweils interessierenden strategischen Dimensionen festgelegt und dokumentiert werden. Der finanzielle Niederschlag der langfristigen und damit meist mehrperiodischen strategischen Planentscheidungen und Maßnahmen wird in strategischen, der von operativen Entscheiden und Maßnahmen wird in operativen Budgets dokumentiert (s. Abb. 5).

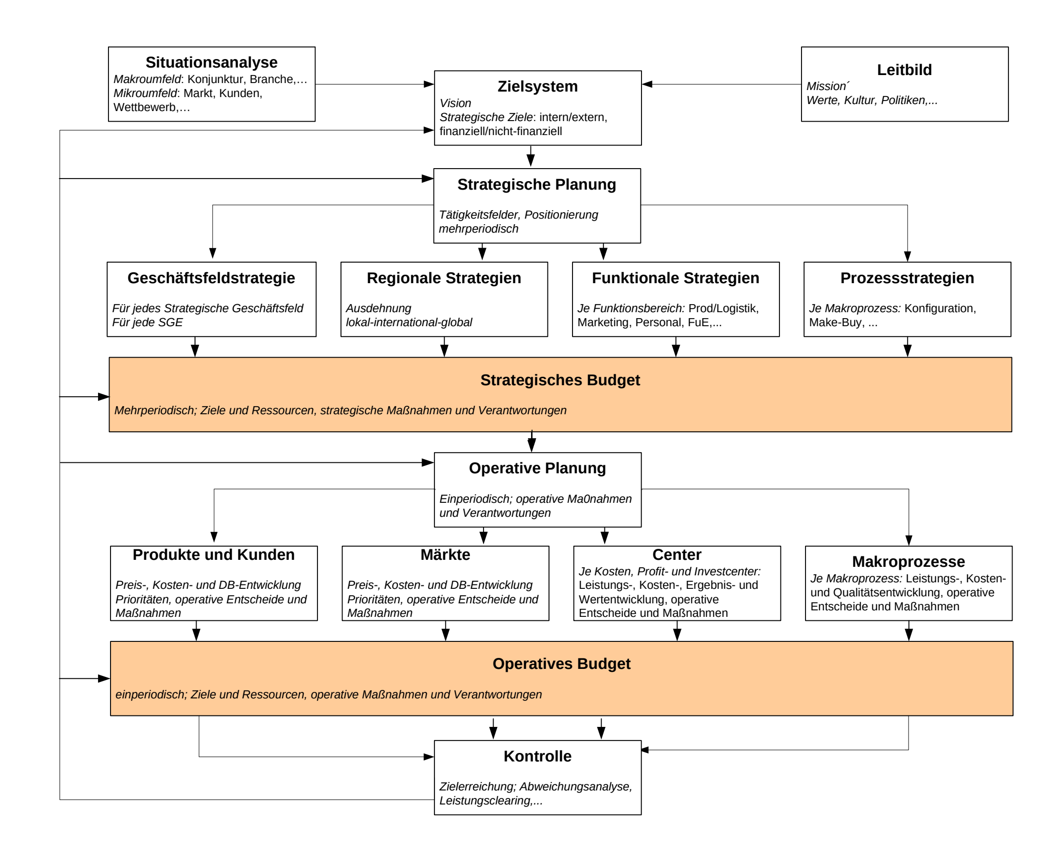


Abb. 5: Strategien und Budgets in einem Planungs- und Kontrollsystem

2.1 Budget

Unter einem Budget wird das formalzielorientierte, in wertmäßigen Größen ausgedrückte Ergebnis des gesamten Planungsprozesses verstanden, der die Ergebniszielorientierung einer Organisation sicherstellen soll. Es legt fest, welche Geldmittel für das Erreichen bestimmter Ziele und die Erledigung der dazugehörigen Aufgaben in den dezentralen Führungseinheiten zur Verfügung stehen. Budgets repräsentieren die finanziellen Einschränkungen bzw. Anreize, die sich aus den Zielsystemen ergeben, die auf die dezentralen Führungsbereiche eines Unternehmens heruntergebrochen werden. Zentrale Aufgabe von Budgets ist damit die Koordination von Entscheidungen über Ziel- und Ressourcenvorgaben für untergeordnete Instanzen (Vorgabeentscheidungen). Budgets besitzen für die betroffenen Verantwortungsbereiche Vorgabecharakter, für dessen Verbindlichkeitsgrad unterschiedliche Ausprägungen anzutreffen sind. Es werden absolut starre, starre und flexible Budgets unterschieden. Bei flexiblen Budgets passen sich wichtige Zielgrößen im Gegensatz zu den starren Budgets automatisch bei bestimmten Veränderungen, wie Beschäftigungsschwankungen, an. Absolut starre Budgets passen sich nicht an Umweltveränderungen an und sind daher ohne Ausnahme einzuhalten.[7]

Die Verbindlichkeiten der Budgets gelten für vorgegebene Fristigkeiten, wobei grundsätzlich gilt, dass je kurzfristiger der Zeithorizont ist, desto detaillierter ist das Budget. Meist besitzen Budgets Fristigkeiten von einem Geschäftsjahr, prinzipiell können Budgets aber für jede Planungsfristigkeit festgelegt werden.

[7] *Brühl, R.*, Controlling (B), 2004, S. 247.

Mehrjahresbudgets sind dabei häufig in stark verdichteter Form anzutreffen. Neben der Unterscheidung in kurzfristige und langfristige Budgets können Budgets in **operative und strategische Budgets** differenziert werden. In der Praxis überwiegen kurzfristige operative Budgets, daneben sind auch eher langfristige strategische Budgets anzutreffen, die zur Aufgabe haben, Strategien formalzielorientiert zusammenzufassen (s. Abb.6). Solche strategischen Budgets setzen einerseits den Rahmen für die detaillierteren operativen Budgets, andererseits sind die in ihnen eingeschlossenen strategischen Maßnahmen mit der operativen Budgetierung zu verzahnen.

Nach der Abhängigkeit von der Bezugsgröße	Nach der Geltungsdauer	Nach den verwendeten Wertgrößen	Nach den Verantwortungs-bereichen	Nach den Verantwortungs-ebenen
Absolut starre Budgets	Unterjährige Budgets	Leistungsbudgets	Funktionsbudgets	Kostenstellen-budgets
Starre Budgets	Jahresbudgets	Kostenbudgets	Profit-Center Budgets	Abteilungsbudgets
Flexible Budgets	Mehrjahresbudgets	Investitionsbudgets	Projektbudgets	Bereichsbudgets
		Finanzbudgets …	Prozessbudgets	Unternehmens-budgets

Abb. 6: Budgetarten

Inhalt eines Budgets ist in erster Linie ein Erfolgsplan, der die zu erreichenden Ziele der Verantwortungsbereiche festlegt und die hierzu aufzuwendenden Ergebnisbelastungen, und nicht ein prognostiziertes Ergebnis der Geschäftstätigkeit. Das Budget als Erfolgsplan besteht aus Erlösen und Kosten, zusätzlich werden Liquiditätsüberlegungen in die Budgets einbezogen werden, wodurch die Verdichtung der Budgets in drei Richtungen möglich wird: budgetierte Erfolgsrechnung, Budget der Finanzmittel und budgetierte Bilanz. Hieraus entstehen inhaltlich integrierte Budgetsysteme, die Vorgaben für alle Grundgrößen des Rechnungswesens liefern und damit die Voraussetzung für eine erfolgs- und wertorientierte Unternehmensführung bilden.

Für die Aufstellung von Budgets können eine Reihe von Grundsätzen definiert werden. Damit die in den Budgets vorgegebenen Plandaten mit möglichst hoher Wahrscheinlichkeit erreichbar sind, sollten Budgets realistisch, umfassend und vollständig sein.[8] Um die Wirksamkeit des Vorgabecharakters von Budgets sicherzustellen, ist ein Soll-Ist-Vergleich zur Kontrolle der Zielerreichung unabdingbar.

2.2 Funktionen der Budgetierung

Unter Budgetierung wird der Managementprozess verstanden, der aus der Aufstellung, Verabschiedung, Kontrolle und Abweichungsanalyse von Budgets besteht.
Diesem Prozess werden in der Unternehmenspraxis eine Vielzahl von Funktionen zugedacht. Folgende Begriffe beschreiben diese Funktionen: Planung, Koordination,

---

[8] *Egger, A../Winterheller, M.*, Budgetierung (B), 2002, S. 59.

Integration, Definition von Leistungsvorgaben, Motivation, Optimierung, Sicherung, Anpassung, Innovation und Kreativität, Ressourcenallokation und -freigabe, Effizienzsteigerung, Prognose, Durchsetzung, Kontrolle, Komplexitätsreduktion, Zielsetzung, Strategieumsetzung, Informationsversorgung, Risikoerkennung, Lernen u.v.m.[9] Im Folgenden werden diese Funktionen der Budgetierung systematisiert.

Die **Prognose** ermöglicht eine realistische Vorausschau auf die Zukunft, was der Identifikation von Chancen und Risiken dient. Prognose wird definiert als Erwartungen über zukünftige Ereignisse, die Beobachtungen oder Theorien als Grundlage haben.[10] Prognosen werden durchgeführt, um die Anpassungsfähigkeit von Unternehmen durch die Erzielung zeitlicher Vorteile zu verbessern. Durch die zukunftsgerichtete Erstellung von Budgets sollen also Probleme (Risiken) und Chancen frühzeitig erkannt und entsprechende Maßnahmen in Gang gesetzt werden.

Die Aufgabe der **Koordination** ist allgemein die gesamtzielgerichtete Durchführung von Teilaufgaben. Sie ist von entscheidender Bedeutung, da die alleinige Ausrichtung auf ein Teilziel zu anderen Entscheidungen führen kann als die gleichzeitige Berücksichtigung des Gesamtziels. Entsprechend werden interdependente Entscheidungen aufeinander abgestimmt. Durch Koordination wird zudem die Komplexität reduziert, da künftiges Verhalten festgelegt wird und daraus der Ausschluss von Handlungsalternativen resultiert. Die Integration von Einzelmaßnahmen soll überdies die Schaffung von Synergien ermöglichen. Hier soll unter Koordination die Abstimmung von Organisationseinheiten auf ein gemeinsames Ziel hin verstanden werden, wodurch die dezentrale Entscheidungsfindung unterstützt werden soll. Zur Koordination soll zudem die Zuteilung bzw. Bewilligung von Mitteln (Ressourcenvergabe) zur Teilzielerreichung gezählt werden. Koordination durch Budgets kann demnach in drei Bereiche eingeteilt werden:

- **Zielvorgaben**: Ziele werden auf die dezentralen Führungseinheiten verteilt. Dies ist ein kaskadierender Vorgang: aus den Oberzielen sind die Zusammenhänge mit den untergeordneten Elementen einer Ziellogik oder –hierarchie zu verdeutlichen sowie die Überlegung transparent zu machen, die zu der Setzung einer bestimmten Zielhöhe geführt hat. Erst durch die Zielvorgabe lassen sich die notwendigen Handlungen der dezentralen Führungseinheiten ableiten.
- **Bereichsabstimmung**: Die Elemente der Ziellogik eines Unternehmens sind deshalb den einzelnen Organisationseinheiten zuzuordnen, deren Realisation wiederum zur Erreichung des Gesamtziels bzw. Unternehmensziels führen. Budgets für untergeordnete Führungseinheiten bedingen deshalb immer auch die Übertragung einer entsprechenden Zielverantwortung.
- **Ressourcenvergabe**: Sie umschließt sowohl die Zuteilung finanzieller als auch operationaler Ressourcen zu den jeweiligen Organisationseinheiten, um die gesetzten Leistungsziele zu verwirklichen.

[9] *Buchner, H.*, Planung im turbulenten Umfeld (B), 2002, S. 61. und *Horváth, P.*, Controlling (B), 2003, S. 168 f. und 206 ff. und *Wall, F.,* Planungs- und Kontrollsysteme (B), 1999, S. 14. und *Grüning, R.*, Planung und Kontrolle (B), 2002, S. 42. und *Pfläging, N.*, Beyond Budgeting (B), 2003, S. 27 und S.494.

[10] *Hansmann, K.-W.*, Kurzlehrbuch Prognoseverfahren (B), 1983, S. 11

Motivation soll plankonformes Verhalten erzeugen, weshalb Motivation an die festgelegten Ziele gekoppelt ist. Durch die Schaffung von (finanziellen) Anreizen, die mit der Zielerreichung verbunden sind, wird versucht, die Organisationsteilnehmer zu zielkonformem Verhalten zu bewegen.

Um zielorientiertes Verhalten der Organisationsteilnehmer zu ermöglichen, ist die **Kommunikation** der Ziele notwendig. Dazu ist die geplante Unternehmensentwicklung in wertmäßigen Größen durch Budgets wiederzugeben. Die Kommunikation findet einerseits bei der Budgetaufstellung statt und andererseits in Form von Reports, die die Budgetvorgaben mit den tatsächlichen Ist-Daten vergleichen.

Die **Kontrolle** dient der Sicherung der Zielausrichtung der getroffenen Entscheidungen und stellt somit einen „Umsetzungscheck" der Planvorgaben von Budgets dar. Aufgabe der Kontrolle ist die Information über erzielte Fortschritte und die Initiierung von korrigierenden Maßnahmen. Wichtiger Aspekt der Kontrolle ist das Lernen aus den getroffenen Entscheidungen. Zudem soll sie, ebenso wie die Motivationsfunktion, zu zielkonformem Verhalten der Entscheidungseinheiten führen.

Somit lassen sich insgesamt sieben Funktionen der Budgetierung definieren:

- Prognose
- Zielvorgabe
- Bereichsabstimmung
- Ressourcenvergabe
- Kontrolle
- Motivation und
- Kommunikation.

2.3 Budgetierungssysteme

Die Budgetierung ist ein komplexer, multipersonaler und vielfach auch ein multilokaler Managementprozess, der einer ablauforganisatorischen und inhaltlichen Struktur bedarf. Letztere wird durch die Festlegung und Beschreibung von Budgetierungssystemen erreicht (s. Abb.7).

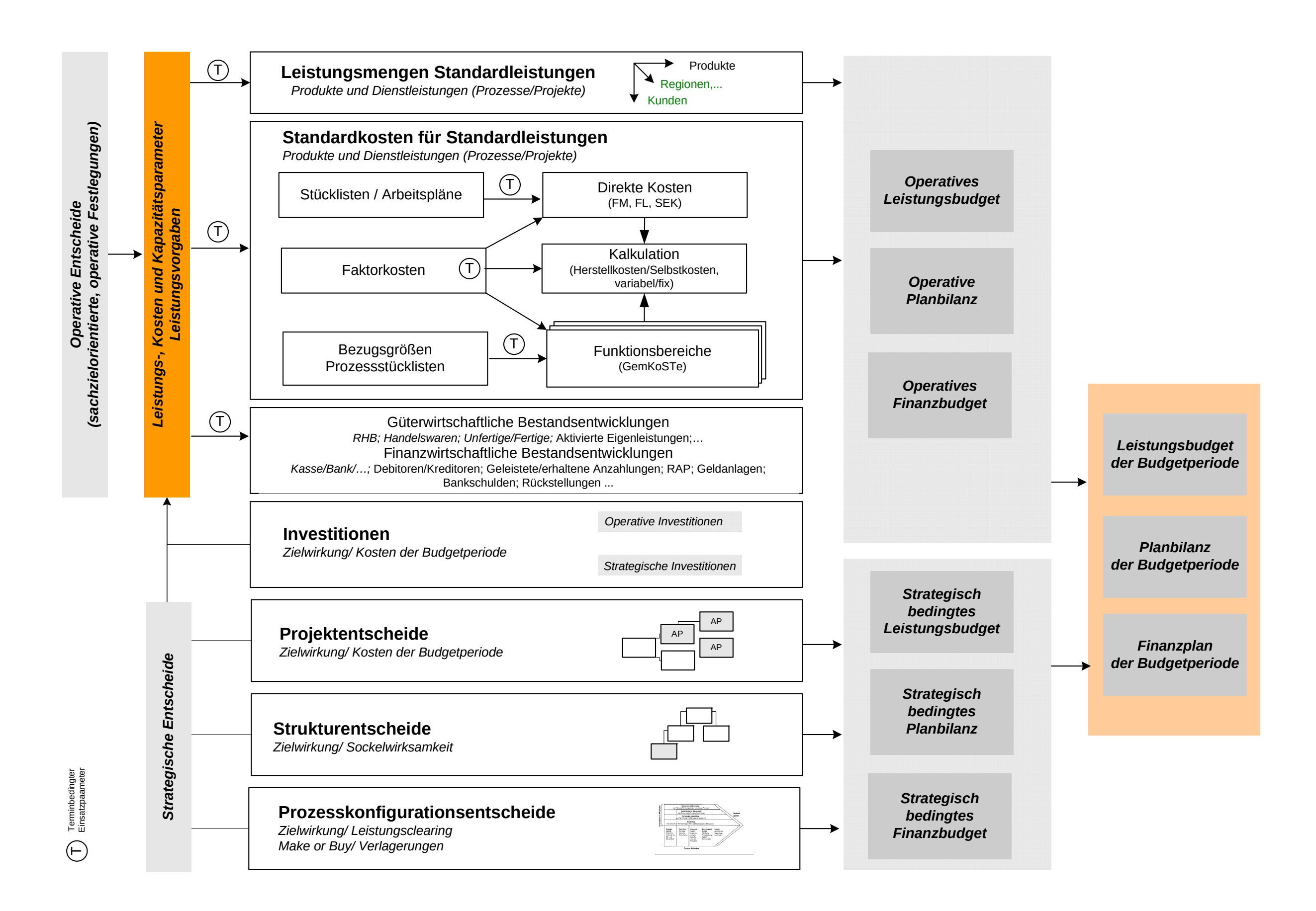


Abb. 7: Budgetierungssystem

Die höchste Aggregationsstufe von Budgetierungssystemen besteht aus dem **integrierten Erfolgs-, Finanz- und Bilanzplan**, der gewissermaßen einen Masterplan des Budgetierungssystems darstellt. Für eine bestimmte Budgetperiode kann dann zwischen dem strategisch und dem operativ bedingten Teil des Budgets unterschieden werden. **Strategische Entscheide**, die eigentlich Teil des strategischen Budgets sind, beeinflussen die Leistungs-, Kosten- und Kapazitätsparameter sowie die Leistungsvorgaben des Budgetierungssystems. Strategische Entscheide können als Umsetzungsentscheidungen für strategische Initiativen Projektentscheide oder Strukturentscheide sein, fließen aber immer nur mit dem Teil in das Budget einer Periode ein, die diese Periode betreffen. Weiterhin können sie als Prozesskonfigurationsentscheide auf Periodenbudgets Einfluss nehmen, indem sie Änderungen in der Wertschöpfungskette (z.B. Entfall von eigenen administrativen Arbeiten, Make or Buy- Entscheidungen, Verlagerungsbeschlüsse an andere Standorte, etc.) einschließen. Für strategische Entscheide können im Budgetierungssystem in der Regel getrennte logische Datenräume eingerichtet werden, so dass sich die Periodenbudgets eines relevanten Führungsbereiches in eine operative und eine strategische Dimension zulegen lassen.

Neben strategischen Entscheidungen gehen operative Festlegungen über Leistungs-, Kosten- und Kapazitätsparameter in die Periodenbudgets ein. Neben den Leistungsmengen und –preisen gehen die Standardkosten und deren relevante Bestimmungsgrößen parametrisch in die operativen Budgets ein. Investitionsprojekte und deren periodische Auswirkungen werden zur Vervollständigung des Leistungsbudgets benötigt. Die güter- und finanzwirtschaftlichen

Bestandsentwicklungen werden überwiegend über Umschlags- oder Verweildauerkennzahlen budgetiert und ermöglichen die Berücksichtigung der entsprechenden Bilanzpositionen. Für bilanz- und steuerpolitische Positionen wird eine Trennung des Effektes von Steuerungsentscheidungen über eine entsprechende Vertikalgliederung des Leistungsbudgets erreicht.

In der Praxis wird der strategisch bedingte Teil eines Periodenbudgets nur einen kleinen Anteil (10-15%) des Gesamtbudgets ausmachen, da Unternehmen die Durchführbarkeit von strategischen Projekten auch an der aggregierten Belastbarkeit einer Erfolgsplanung für das Kerngeschäft (also die Standardleistungen) verproben müssen, um den periodischen Leistungserwartungen der Stakeholder entsprechen zu können.

2.4 Budgetierung als Führungsinstrument

In Unternehmen wird die Budgetierung als primäres Managementsystem zur Zielfestlegung, Ressourcenallokation und Schaffung von Leistungsanreizen genutzt. Häufig ist es das alleinige Instrument der erfolgsorientierten Unternehmenssteuerung.[11] Das Unternehmen wird mit Hilfe der Vollständigkeit und Detaillierung der Budgetierung exakt abgebildet und bietet entsprechend erschöpfende Informationen für Analysezwecke.[12] Diese vielseitigen Informationen und Analysezwecke dienen zum Beispiel zur Erläuterung der Abweichungen, zur Vorausschau zukünftiger Abweichungen, zum Entwickeln von Maßnahmen, welche künftige negative Abweichungen vermeiden und vermindern helfen, und zum Erkennen von Folgewirkungen der Abweichungen und zu deren Gegensteuerung. Zudem können anhand der zahlreichen Informationen und Analysemöglichkeiten Potentiale und Schwachstellen definiert werden, und man erkennt Kostenbelastungen ex-ante. Mit Hilfe der Budgetierung können ebenso Ziele konkretisiert und der Gewinnbedarf ermittelt werden. Das Budget ermöglicht auch eine quantifizierte Koordination der Teilpläne.[13] Die Budgetierung dient in den meisten Unternehmen seit langem als zentrales Instrument zur Prognose, Koordination und Motivation.

Überdies wurde die Prognose als Budgetfunktion identifiziert. Zwar kann sie nicht als Steuerungsfunktion bezeichnet werden, da sie keinen gestaltenden Charakter besitzt,[14] jedoch ermöglicht sie eine realistische Vorausschau auf die Zukunft und kann somit die Grundlage für die Wahrnehmung von Problemen sein, die anpassende Entscheidungs- und Planungsakte in Gang setzen kann. Die Umformung der Prognoseinformationen in Entscheidungen erfolgt in der Planung, die Teil des Führungsprozesses ist.

Dadurch wird deutlich, dass der gesamte Führungsprozess entscheidend durch die Ausgestaltung einer effektiven und effizienten Budgetierung begleitet wird. Somit ist die Budgetierung als ein wichtiges Steuerungsinstrument anzusehen. Auch Kritiker der Budgetierung teilen die Auffassung, dass Budgets und Budgetierung die Steuerungsbedarfe des Managements erfüllen und den vorrangigen Managementprozess zur Zielfestlegung, Ressourcenallokation und

[11] *Gleich, R./ Kopp, J./ Leyk, J.*, Ansätze zur Neugestaltung, 2003, S. 461.
[12] *Weber, J./ Linder, S.*, Budgeting, 2003, S. 12.
[13] *Peemöller, V.*, Grundlagen und Einsatzgebiete des Controlling, 1997, S. 162.
[14] *Grünig, R.*, Planung und Kontrolle (B), 2002, S. 23.

Leistungsüberwachung darstellen sollen.[15] Pfläging definiert Budgetierung dementsprechend als alle mit dem Budget verbundenen Prozesse des Leistungsmanagements. Hier wird erneut deutlich, dass die Erarbeitung und Aufstellung der Budgets genauso zur Budgetierung gehört wie ihre Wiederverwendung in der Leistungsmessung, Kontrolle und Abweichungsanalyse.
Es drängt sich die Frage auf, ob eine solche zentrale Stellung inklusive der Funktionsbreite, die Budgets in der Welt der betriebswirtschaftlichen Steuerung haben sollen, nicht zu einer Überforderung dieses Instrumentes führen müssen. Trifft dies zu, ist eine Entfeinerung der Budgetierung und eine funktionale Desintegration angesagt.

Die Steuerungsfunktion von Budgets soll die Umsetzung der Strategien und der relevanten Maßnahmen bewirken. Dabei führt der Zweck der Koordination dazu, dass Manager bei der Festsetzung der Zielhöhe realistische Werte ansetzen sollten. Gleichzeitig führt die mit der Budgetierung einhergehende Verantwortungsdelegation zu der Tendenz, dass die dezentralen Bereiche die einmal gesetzten Budgets auch tatsächlich ausschöpfen, beispielsweise durch eine nicht zielführende und ineffektive Mittelverwendung. Und letztlich: die Funktion, der Maßstab für die spätere Leistungsbeurteilung oder gar Leistungsbelohnung zu sein, führt dazu, dass das Interesse des dezentralen Bereiches an einem möglichst niedrigen Leistungsziel hoch ist. Die Transparenz schaffenden Ausgleichsmaßnahmen des Controllings sind für die einzelnen Führungsphasen in Abbildung 8 zusammengestellt. In diesem Sinn dient die Funktion des Controllings also der Rationalitätssicherung der Führung im Sinne von Jürgen Weber vorgetragenen Controllingkonzeption[16].

[15] *Pfläging, N*., Beyond Budgeting (B), 2003, S. 24 ff.
[16] *Weber, Jürgen; Schäffer, Utz; Weber [Einführung 2008]*

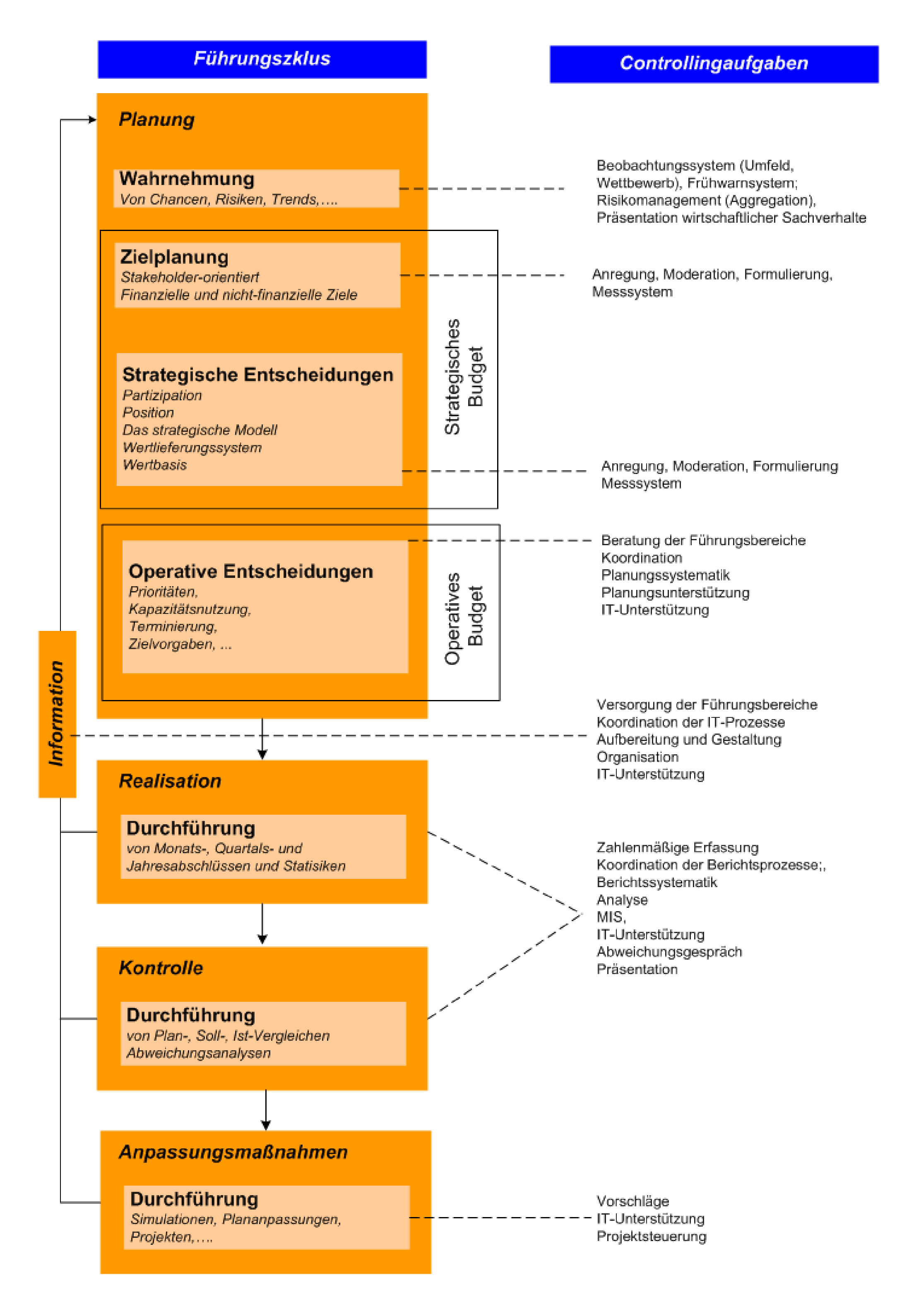


Abb. 8: Budgetierung im Führungsprozess

Das Dilemma einer hyperfunktionalen Budgetierung zeigt sich in der Praxis in einigen zum Teil gravierenden Dysfunktionalitäten der Budgetierung bei der Steuerungswirkung:

- **Mangelhafte Verknüpfung zwischen Strategie und Budget:** Bei der Ressourcenzuteilung konzentriert man sich auf operative Fragen, strategische und operative Planung wird in verschiedenen Welten mit abweichenden Methoden und Verantwortlichkeiten geregelt, ein systematischer Übergang der Strategie in Maßnahmen und Budgets unterbleibt.
- **Strukturkonservatismus:** Die Budgetierung lehnt sich zu stark an den vorhandenen Strukturen und Erfahrungswerten an. Eine dynamische Verbindung von Leistungsmenge und Ressourcenzuteilung unterbleibt. Ebenso ist die klassische Budgetierung input- und vergangenheitsorientiert, denn zum einen werden oft nur Vergangenheitswerte fortgeschrieben, und es erfolgt dabei nur teilweise oder gar nicht die Berücksichtigung von Ineffizienzen, zukünftigen Herausforderungen und Innovationen. Zum anderen liegt die Konzentration bei der Erstellung nur auf der Rechtfertigung von Ressourcenverwendungen anstatt auf der Bewertung der Ausbringungen.[17]
- **Fehlende Reagibilität auf externe Veränderungen, besonders des Marktes:** Der Jahreszyklus einer klassischen Budgetierung ist länger als viele Marktbewegungen und –zyklen. Chancen neuer Geschäftsmöglichkeiten werden so verpasst, andererseits besteht das Risiko, dass die Kostenstrukturen nicht schnell genug an die Marktänderungen angepasst werden.
- **Anreiz- und Motivationswirkungen auf Abwegen:** Bei der traditionellen Budgetierung erfolgt die Ableitung der Ziele aus einer internen unternehmensorientierten Sichtweise, welche nicht zwangsläufig mit den Anforderungen des Marktes konform ist. Es besteht die Tendenz, zu niedrig angesetzte Zielniveaus (moderat) überzuerfüllen oder bestehende Budgets pro forma auszugeben und so auf neue Rechnung vorzutragen.

Zusätzlich zu diesen dysfunktionalen Aspekten der Budgetierung wird von der Unternehmenspraxis kritisch angemerkt, dass die Budgetierung nur mit großem personellen Aufwand und umständlichen, unvollständigen Abstimmungsprozessen erstellt werden könne. Der Ressourcenverbrauch für den Prozess der Budgetierung sei zu hoch und sollte besser für wertschöpfende Prozesse eingesetzt werden.[18] „Experten schätzen den Aufwand für Planung und Budgetierung im Management auf 10 bis 20 Prozent der Arbeitszeit, im Controlling auf über 50 Prozent.“[19]

Trotz der dominierenden Kritikpunkte kam man durch Studien und Schätzungen zu dem Ergebnis, dass sich 90-99% der Unternehmen in Europa und den USA bei der Unternehmenssteuerung weitgehend in einem geringeren oder höheren Maße auf die traditionelle Budgetierung verlassen. Eine Ursache für diese Sonderstellung kann die Haltung des Managements gegenüber eingeführten Steuerungsinstrumenten sein. Opportunistisch handelnde Manager favorisieren finanzielle, fixe und periodenorientierte Pläne, weil die Budgets sie ihnen das Gefühl einer sicheren, überraschungsarmen Vorausschau in die Zukunft vermitteln. Die Rituale der Budgetierung sind so tief in vielen Organisationen und Führungsebenen verankert,

---

[17] *Amrein, S./ Widmer, M./ Witmer, D.*, Better Budgeting, 2003, S. 269.
[18] *Becker, A.,* Beyond Organization, 2004, S. 82.
[19] *Gaiser, B./ Gleich, R.*, Die Evolution ist machbar, 2004, S. 26.

dass man sie nur mit großer Mühe im Rahmen von Change Management Initiativen beheben kann.[20]

3 Empfehlungen zur Weiterentwicklung der Budgetierung

3.1 Impulse aus der Beyond Budgeting Initiative

Jeremy Hope, Buchautor und Ex-Manager, und Robin Fraser, Managementberater, waren aufgrund der in dem vorangegangenen Kapitel bereits erläuterten Problemfelder schon sehr frühzeitig der Meinung, dass eine effizientere und effektivere Unternehmenssteuerung nicht in der Verbesserung der Budgetierung zu finden sei, sondern dass die Budgetierung völlig abgeschafft werden müsse. Als bekannt wurde, dass im skandinavischen Raum erfolgreich ohne Budgets gearbeitet wird, riefen Hope und Fraser gemeinsam mit der CAM-I (Consortium for Advanced Manufacturing – International) 1997 ein Forschungsprojekt ins Leben, das sich mit der Forschung nach einem neuen Managementmodell für den Übergang vom Industriezeitalter zum Informationszeitalter beschäftigte. Jeremy Hope, Robin Fraser und Peter Bunce, der zuvor das europäische Büro des amerikanischen CAM-I in London geleitet hat, schlossen sich zusammen und gründeten 1998 den Beyond Budgeting Round Table (BBRT) in London. Der BBRT ist eine mitgliederfinanzierte Organisation zur Erforschung und Weiterentwicklung des Unternehmenssteuerungsmodells Beyond Budgeting. Seit 1998 verbucht die Organisation mehr als 70 Mitglieder und hat circa 20 Fallstudien erfolgreicher Beyond Budgeting Umsetzungen erstellt. Die Fallstudien werden den Mitgliedern zur Verfügung gestellt. Im Jahre 2003 hat sich der BBRT vom CAM-I losgelöst und operiert als ein unabhängiges Netzwerk mit Schwester-Beyond Budgeting Round Tables in Australien und den USA. Der BBRT hat sich mit seiner Arbeit verschiedene Ziele gesetzt, wie zum Beispiel die Untersuchung von Unternehmen, die ohne Budgets managen, und die Erstellung entsprechender Case Studies. Auf der Grundlage dieser Case Studies sollen die dem Führungs- und Steuerungssystem zugrunde liegenden Prinzipien extrahiert, allgemein entwickelt und in Form von Guidelines beschrieben werden. Eine weitere Zielsetzung des BBRT ist die ständige Weiterentwicklung des Beyond Budgeting Konzepts, die Bereitstellung von Beratungs- und Diskussionsplattformen für die Mitglieder (shared learning) und die Erhöhung der Zahl der Beyond Budgeting- Umsetzungen.[21]

Es wurde bereits gezeigt, dass es einen engen Zusammenhang zwischen den Veränderungen des Unternehmensumfeldes, den neuen kritischen Erfolgsfaktoren und den daraus folgenden Anforderungen an das Führungssystem von Unternehmen gibt. In diesem Lichte kann man das Beyond Budgeting- Konzept als ein integriertes Managementmodell verstehen, dass den aktuellen Herausforderungen des Unternehmensumfeldes gerecht werden will. In diesem Managementmodell finden zwei wesentliche Gestaltungselemente Anwendung.

[20] *Pfläging, N.*, Fundamente des Beyond Budgeting, 2003, S. 189.
[21] *Daum, J.*, Ein Management- und Controlling-Modell für nachhaltigen Unternehmenserfolg, 1999, S. 2/411.

Das erste Gestaltungselement sind neue Management- und Führungsprinzipien, die auf einer radikalen Dezentralisierung („Devolution“) von Entscheidungen basieren. Bei dieser neuen Führungsvision erfolgt vor allem die Delegation von Entscheidungen auf niedrigere Führungsebenen. Die Führungsvision ist geprägt durch marktnah operierende Mitarbeiter, die zur Verantwortungsübernahme motiviert sind und denen es erlaubt ist, selbstständig zu entscheiden. Den Mitarbeitern wird der Zugriff auf benötigte Ressourcen zu den Zeitpunkten gewährleistet, zu denen sie von ihnen benötigt werden. Am Ende dieses Prozesses würde eine netzwerkartige, weitgehend hierarchiefreie Organisationsform stehen.

Das zweite Gestaltungselement sind hochgradig flexible, adaptive Managementprozesse. Diese gestatten das markt- und kundenorientierte Handeln. Im Gegensatz zur traditionellen Budgetierung basieren die Managementprozesse nicht auf fixen Zielen und Plänen, sondern ermöglichen laufende Anpassungen an Umfeldbedingungen und Kundenanforderungen. Diese beiden Dimensionen des Beyond Budgeting Konzepts werden durch 12 Beyond Budgeting Prinzipien näher erläutert. Sechs Prinzipien beziehen sich auf das Führungssystem und die anderen sechs Prinzipien auf das Steuerungssystem. Abb.9 zeigt das Beyond Budgeting Konzept mit seinen zwei Gestaltungselementen und den jeweils sechs Prinzipien.

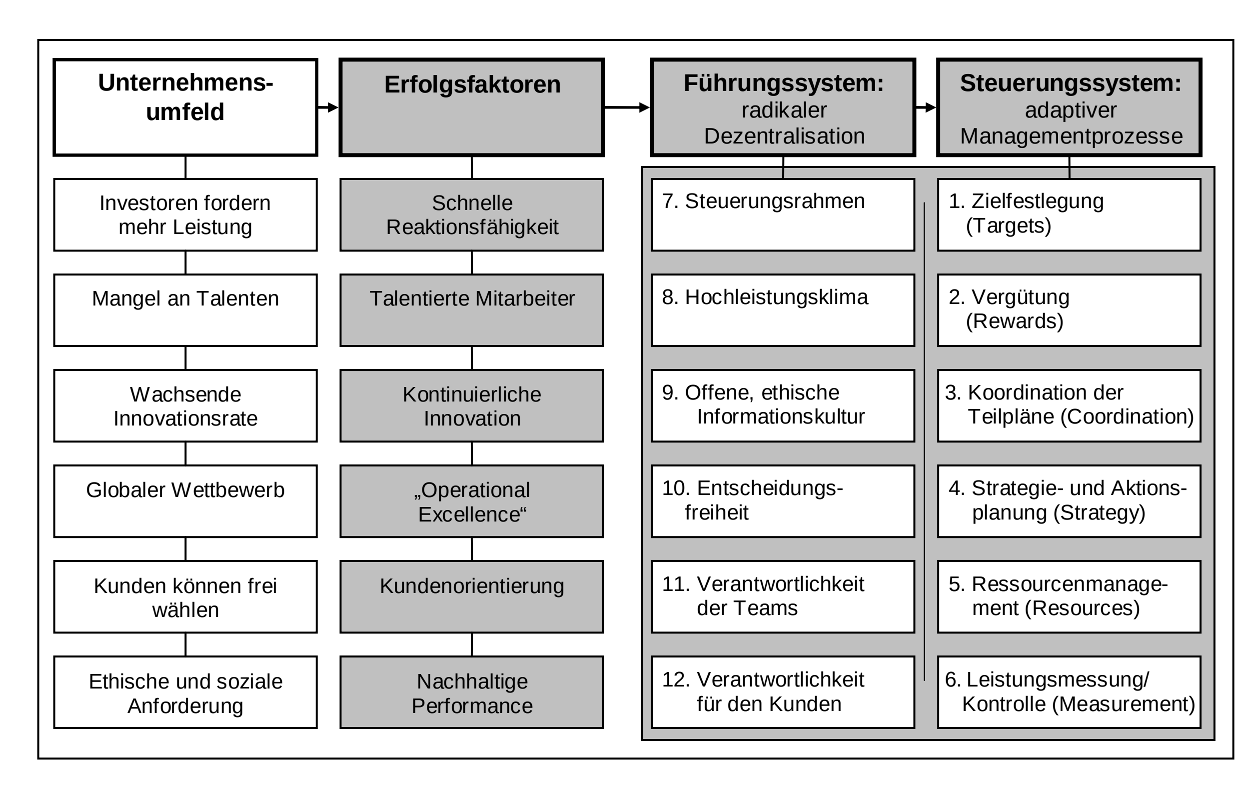


Abb. 9: Gestaltungsprinzipien des Beyond Budgeting Konzepts

Quelle: In Anlehnung an *Weber, J./ Linder, S.*, Budgeting, 2003, S. 14.

In dem hier interessierenden Zusammenhang sind besonders die 6 Gestaltungsprinzipien adaptiver Managementprozesse von Belang:

- Es soll eine permanente Überprüfung und Weiterentwicklung der Strategie erfolgen.
- Die Ziele im Unternehmen sollen laufend gegen externe Benchmarks adjustiert werden und sich in der Zielhöhe deshalb kontinuierlich verändern.
- Die Maßnahmenplanung soll an Strategieänderungen und neue Markterfordernisse laufend angepasst werden.
- Der aktuelle Bedarf soll eine flexible Ressourcensteuerung bewirken.
- Es wird eine offene Kommunikation über die eigene, in Zahlen gemessene Leistung und die der Vergleichsgruppe erfolgen.
- Leistungsbeurteilung und monetäres Anreizsystem unterliegen dem ex-post-Vergleich der eigenen Ist-Ergebnisse mit denen der Vergleichsgruppe.

Diese sechs strikt formulierten Prinzipien zeigen, dass im Beyond Budgeting-Konzept eine ausgeprägte Ziel- und Zahlenorientierung gesucht wird. Dieses Konzept unterstellt dezentrale Entscheidungsautonomie eigenverantwortlicher Teams, ein Klima der Leistungsorientierung einhergehend mit der Bereitschaft, sich offen messen und vergleichen zu lassen. Eigentlich scheint ein solches Budgetierungs- oder Planungsumfeld zutreffend für Unternehmensberatungen kennzeichnend zu sein, ein Berufsstand, der stark im BBRT vertreten ist. Wenn man in der Frage der Anwendbarkeit der Beyond Budgeting- Konzeption auf generelle Anwendungscharakteristika abzielt, dann ist dieses Umfeld durch hohe Dynamik und Komplexität, also eine hohe Turbulenz gekennzeichnet. Abb.10 greift auf die Forschung von Buchner in diesem Zusammenhang zurück. Buchner kommt zu dem Ergebnis, dass man in turbulentem Umfeld ein besonderes Planungssystem benötigt, dass er mit Rückgriff auf Mintzberg als „Innovative Adhocratie“ bezeichnet und dessen Beschreibung sehr den Gestaltungsprinzipien des Beyond Budgetings entspricht. Planung hat in der Innovativen Adhocratie nur einen mittleren Stellenwert, deshalb kann man auf sie in diesem Umfeld verzichten, wenn man hierfür andere Koordinationsmechanismen wie Zielplanung und –steuerung, direkte Ressourcensteuerung, Business Pläne und ähnliches einsetzt.

Was für den Turbulenzquadranten gilt, gilt aber nicht für die anderen Felder der Dynamik-Komplexitätsmatrix. Insbesondere komplexe Unternehmensstrukturen werden auf die Koordinationswirkung einer die Budgetierung nutzenden Planung nicht verzichten wollen. Hiermit ist auch zu erklären, warum einige größere Organisationen, die zunächst im Zuge der Einführung des Beyond Budgeting auf eine Budgetierung verzichtet haben, später zu ihr zurückgekehrt sind.

Bei allen Zweifeln an der Generalisierbarkeit und der Umsetzbarkeit des Beyond Budgeting gehen von dieser Initiative aber Impulse aus, die bei der materiellen Ausprägung von Planungs- und Budgetierungssystemen berücksichtigt werden sollten:

- Strategie ist als ganzheitlicher und kontinuierlicher Managementprozess zu verstehen.
- Budgetierung und Anreizsysteme sollten entkoppelt werden, um die Politisierung der Zielniveaufindung aus der Budgetierung zu eliminieren.

- Aus diesem Grund ist auch die Einführung relativer Ziele zu befürworten und die Bedeutung von Benchmarks zu unterstreichen.
- Rolling Forecasts sollten als permanente Planadjustierung genutzt werden.

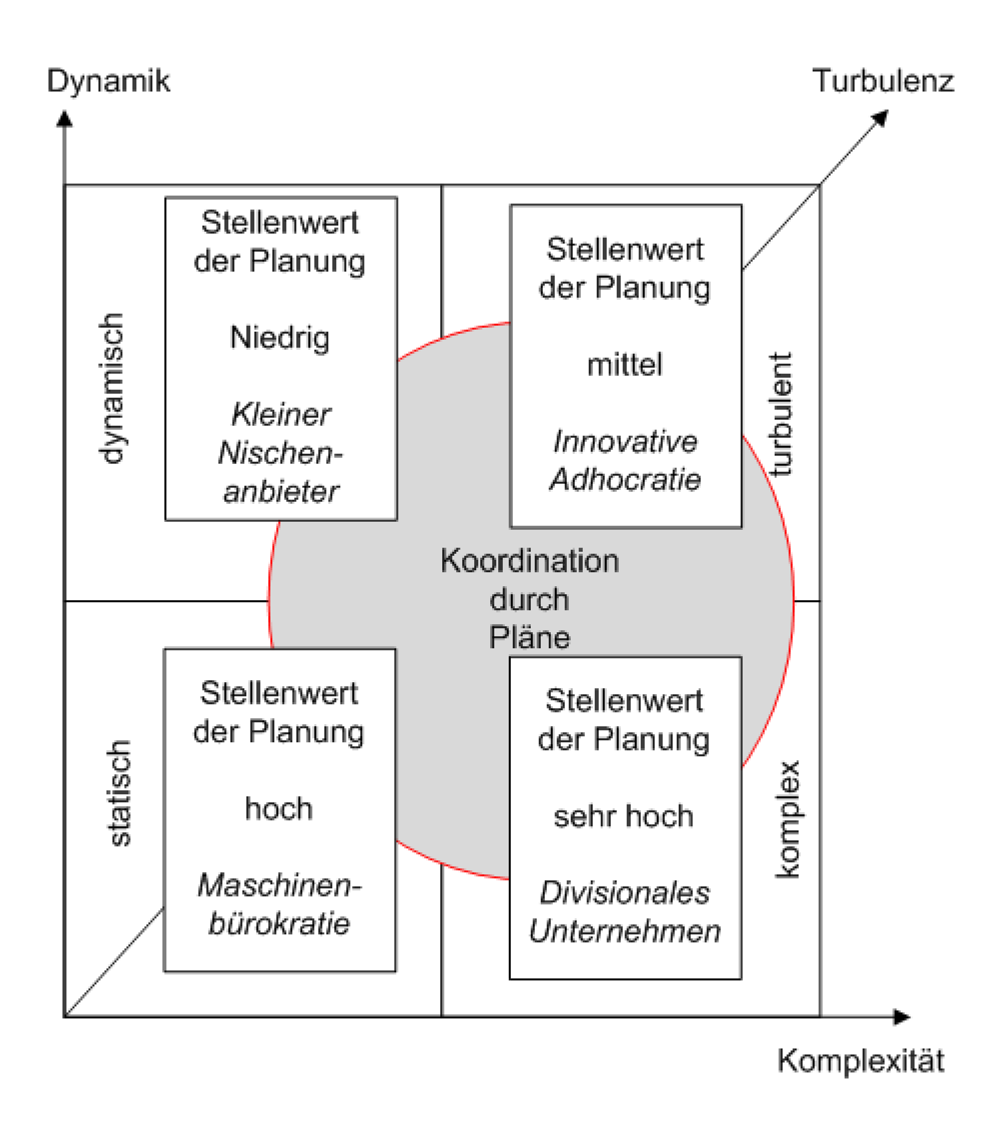


Abb. 10: Planung in turbulentem Umfeld

Quelle: in Anlehnung an Buchner, Holger: "Planung im turbulenten Umfeld – Konzeption idealtypischer Planungssysteme für Unternehmenskonfigurationen", Vahlen Verlag, München 2002, S. 129

3.2 Strategieinduktion in die Budgetierung

Bislang wurde aufgezeigt, dass die Integration der Strategien in die Budgetierung ein Merkmal moderner Planungssysteme ist. Die bewusste Verfolgung einer Strategie zur Erreichung einer Vision erfordert, dass der Ressourceneinsatz explizit an ihr auszurichten ist. Allerdings stehen sich der kurzfristige, meist auf das Geschäftsjahr bezogene Budgetierungszyklus und die langfristige Orientierung der Strategien gegenüber. Ein Instrument, das sich die Integration von Strategien, ihre

Ressourcenausstattung und eine kurzfristige, meist kalenderperiodische Budgetierung zum Ziel gesetzt hat, ist die Balanced Scorecard.[22]

Bei der Balanced Scorecard (BSC) handelt es sich idealiter um ein Managementkonzept für ein strategieorientiertes Leistungsmanagement. Entwickelt wurde es Anfang der 90er Jahre von Robert S. Kaplan und David P. Norton. Sie berücksichtigten bei ihrer Konzeption die vorherrschende Kritik, dass Managementsysteme eine zu starke finanzielle Ausrichtung besitzen. Stattdessen wurde ein ausgewogenes Set an finanziellen und nicht-finanziellen Messgrößen entwickelt, um die gesamte Wertschaffung eines Unternehmens angemessen zu bewerten. Grundgedanke ist, dass zur Leistungsbewertung unterschiedliche Leistungsfelder relevant sind, wie z.B. Finanzen, Kunden oder Prozesse. Heute ist BSC ein weit verbreitetes System zur strategieorientierten Leistungssteuerung.

Aus der Unternehmensvision werden die erfolgskritischen Ziele der Unternehmung abgeleitet, die Gegenstand der Balanced Scorecard sind.[23] Entscheidend für diese Ziele ist ihr Strategiebezug und die Fokussierung auf die bedeutsamen Determinanten. Zudem sollen die Ziele ein ausgewogenes Bild der Strategie liefern und deshalb nicht ausschließlich auf finanzielle Messgrößen konzentriert sein, sondern auch nicht monetäre Messgrößen beachten, die das finanzielle Ergebnis beeinflussen. Zur Unterstützung dieses ausgewogenen Bildes werden die Ziele unterschiedlichen Perspektiven zugeordnet. Diese Perspektiven sind standardmäßig die Finanz-, Kunden-, Prozess- und Potenzialperspektive. Diese stehen in einer hierarchischen Beziehung zueinander, die Auskunft darüber gibt, „...welche Kenntnisse, Qualifikationen und Systeme Beschäftigte brauchen werden [(Potenzialperspektive)], damit das Unternehmen innovativ sein sowie die strategisch richtigen Kapazitäten und Leistungen aufbauen kann [(Prozessperspektive)], um marktgerechte Wertangebote machen zu können [(Kundenperspektive)], die letztlich zu einem höheren Shareholder-Value führen [(Finanzperspektive)].“[24] Die strategischen Ziele stehen also in einem Ursache-Wirkungs-Zusammenhang, der die inhärente Logik der Strategie deutlich macht. Die Darstellung der strategischen Ziele und ihrer Ursache-Wirkungsbeziehungen werden Strategy Maps genannt, sie stellen das strategische Modell einer Strategie dar.

Im Zusammenhang mit der Budgetierung eignet sich die Balanced Scorecard einerseits zum Controlling von Maßnahmen und andererseits lassen sich aus ihr Implikationen für die Budgets ableiten.

3.2.1 Maßnahmencontrolling

Kaplan und Norton schlagen vor, die Strategie durch ein Step-Down- Verfahren mit dem Budget zu verbinden. Erster Schritt ist die Übersetzung der Strategie in eine Balanced Scorecard. Die Ziele der Balanced Scorecard repräsentieren das zukünftig angestrebte Anspruchsniveau, um die Strategie erfolgreich umzusetzen. Das Anspruchsniveau ist im zweiten Schritt festzulegen. Ist dieser Anspruch nicht ohne Weiteres erreichbar, sind drittens Aktionsprogramme für die Erreichung der Ziele zu

[22] Vgl. *Horváth & Partners* (Hrsg.), Beyond Budgeting umsetzen (B), 2004, S. 124 ff.
[23] *Horváth & Partner* (Hrsg.), Balanced Scorecard umsetzen (B), 2001, S. 10.
[24] *Kaplan, R./Norton, D.*, Strategiekarten (A), 2004, S. 56.

identifizieren (vgl. Abbildung 11).[25] Diesen werden die benötigten Finanz- und Personalressourcen zugewiesen. Die Autoren schlagen einen Zeitraum von zwei bis drei Jahren für den Zukunftsbezug der Ziele und die Laufzeit der Programme vor, was dem Unternehmen ermöglicht, die Steuerung anhand strategischer Themen durchzuführen.[26]

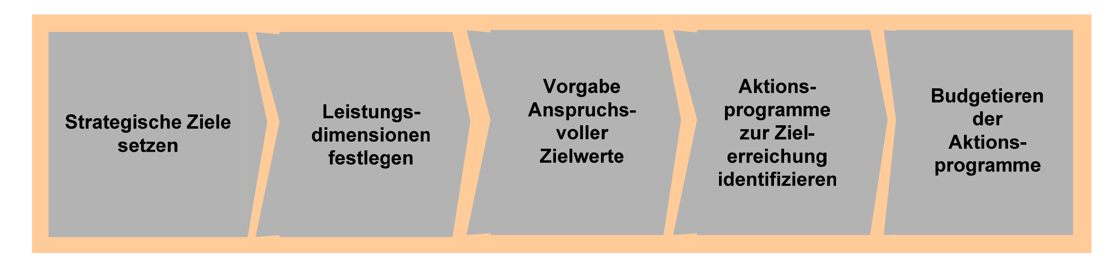


Abb. 11: Budgetentscheide für Aktionsprogramme

In die Jahresbudgets werden die Aktionsprogramme integriert, indem sie auf Maßnahmen heruntergebrochen werden, die in ihrer Gesamtheit dazu beitragen sollen, mittel- und langfristige strategische Ziele zu verwirklichen. Der Ressourcenbedarf dieser Maßnahmen wird budgetiert, und die Maßnahmen erhalten Outputziele, um ihren Beitrag zur strategischen Zielerreichung bewerten zu können. Dadurch bewegt sich die Budgetierung von einem bottom-up-getriebenen Verfahren zu einer Planung mit stärkerem Zielcharakter, in deren Mittelpunkt die Frage steht, was erreicht werden soll, und nicht, was erreicht werden kann. Die Zielvorgaben der Balanced Scorecard verknüpfen somit strategische und operative Planung. Abb. 12 macht deutlich, wie Ergebnisziellücken durch Maßnahmen geschlossen werden können. Das beschriebene Prinzip stellt allerdings nur einen Orientierungsrahmen dar, da es sich in der Realität nur selten umsetzen lassen wird.[27]

[25] *Kaplan, R./Norton, D.*, Die strategiefokussierte Organisation (B), 2001, S. 259.
[26] *Kaplan, R./Norton, D.*, Die strategiefokussierte Organisation (B), 2001, S. 248 f.
[27] *Horváth & Partners* (Hrsg.), Beyond Budgeting umsetzen (B), 2004, S. 135.

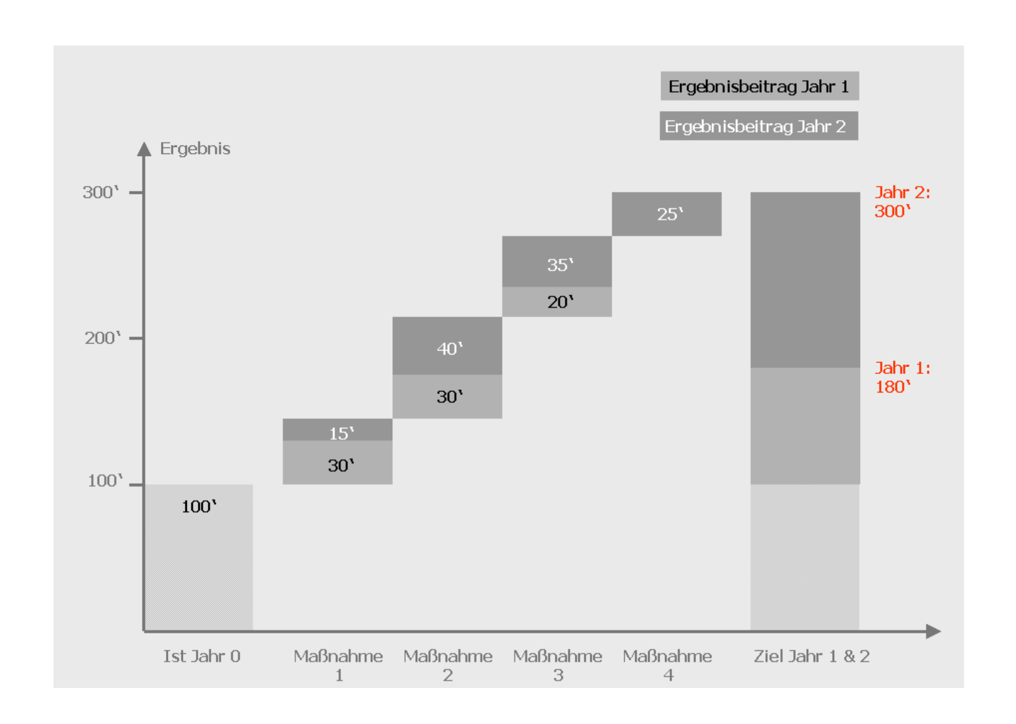


Abb. 12: Ergebnisbeitrag von Maßnahmen

Quelle: ähnlich *Horváth und Partners (Hrsg.)*, Beyond Budgeting umsetzen (B), 2004, S. 135.

Durch die Balanced Scorecard finden strategische Aktionen Eingang in das operative Controlling, das zur Sicherstellung der Aktions- und damit Strategieumsetzung ein permanentes Maßnahmencontrolling im Zuge des operativen Steuerungsprozesses durchführt. Typischerweise werden diese Maßnahmen in Projekten realisiert, weshalb ein gut funktionierendes Projektcontrolling wichtige Voraussetzung für die Umsetzung von Strategien ist. Das Projektcontrolling muss sowohl die Kontrolle der Meilensteine als auch der Ergebniswirkung vereinen,[28] wobei als Ergebnis des Projektes das strategische Ziel aufzufassen ist, zu dessen Erreichung die Maßnahme ins Leben gerufen wurde. Dieser Ansatz steht in enger Verbindung mit der Budgetierung von Projekten, die im Verlaufe dieser Arbeit bereits ausführlich erläutert wurde.

Das Ergebnis dieses Prozesses ist ein strategisches Budget, das die strategischen Aktionsprogramme beinhaltet, welche Investitionen in neue Produkte, Dienstleistungen, Fähigkeiten, Prozesse etc. darstellen. Jedoch umfasst dieses Budget lediglich zehn Prozent des gesamten Unternehmensbudgets, da der Großteil der Unternehmensressourcen durch das bestehende Basisgeschäft beansprucht wird.[29] Vorteilhaft ist, dass die abgeleiteten Aktionsprogramme aus der Summe der Aktivitäten deutlich hervorgehoben und strategische Themen priorisiert mit Ressourcen ausgestattet werden.

Ein Planungssystem, das die strategische mit der operativen Planung über strategische, auch nicht-finanzielle, Ziele und Aktionen sowie dessen

[28] *Horváth & Partners* (Hrsg.), Beyond Budgeting umsetzen (B), 2004, S. 134 f.
[29] *Kaplan, R./Norton, D.*, Die strategiefokussierte Organisation (B), 2001, S. 255 f.

Maßnahmencontrolling verknüpft und dessen Zukunftsbezug durch Forecasts sicherstellt, kann derzeit als State-of-the-Art bezeichnet werden.[30]

3.2.2 Implikationen für die Budgetierung

Durch die Strategy Map und die damit verbundenen Zielindikatoren wird deutlich, welche Kundengruppen, Differenzierungsmerkmale oder Prozesse zukünftig im Zentrum des Steuerungssystems des Unternehmens stehen. Werden strategischen Zielen keine Aktionen oder keine vom aktuellen Ist-Zustand abweichenden Zielwerte zugedacht, ist hieraus auch ersichtlich, dass der Status quo der Aktivitäten, die zu diesen Zielen in Verbindung stehen, zufriedenstellend ist und keiner wesentlichen Veränderungen bedarf. Dies ist eine wichtige Information für den Prozess der operativen Planung und der Budgetierung: Folglich kann bei diesen Aktivitäten auf Basis der bekannten Strukturen und Prozesskonfigurationen geplant werden. Wird das Erreichen von strategischen Zielniveaus durch die Zuordnung von Maßnahmen oder Aktionen unterstützt, dann bedeutet dies, dass das Wertlieferungssystem (Struktur, interne Prozesse, Input, Output, externe Wertlieferung,…) oder die Wertbasis (immateriell: Human-, Wissens- oder Organisationskapital; materiell: Finanzressourcen, Infrastruktur) verändert werden müssen, um das erwünschte Zielniveau erreichen zu können. Dies geschieht in Form von Projekten, die geplant werden müssen und mit diesen Planungen leicht in die Budgetierung eines Kalenderzeitraumes zu integrieren sind. Allerdings gehen Horvath&Partner davon aus, dass dieser projektgebundene strategische Teil kalenderzeitbezogener Budgets nur etwa 10% ihrer Gesamtumfänge ausmachen, und weitere 20% der Budgets für Ressourcen ausgegeben werden, deren Leistung nahe an der Erfüllung strategischer Ziele anzusiedeln ist.

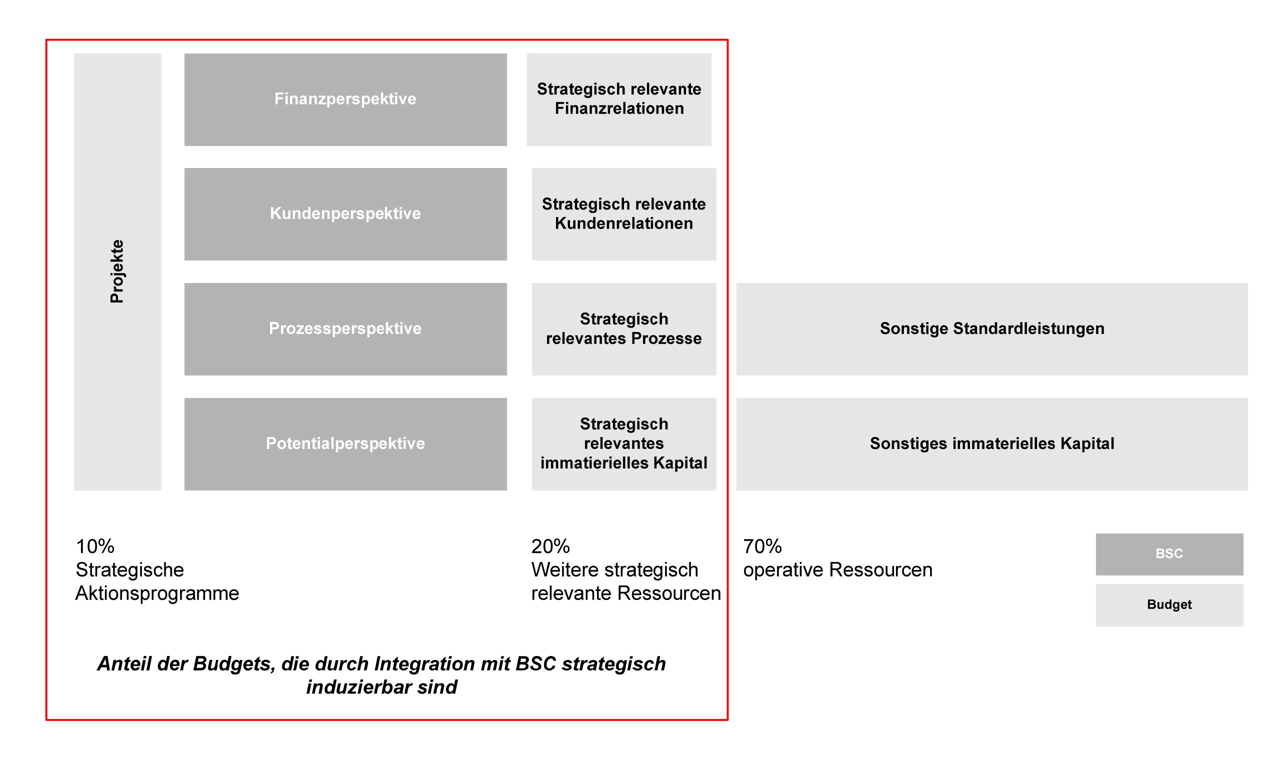


Abb. 13: Zusammenspiel von Budgetierung und Balanced Scorecard

[30] *Horváth & Partners* (Hrsg.), Beyond Budgeting umsetzen (B), 2004, S. 52.

Somit beeinflusst die Balanced Scorecard nicht nur zehn Prozent des Budgets, sondern ermöglicht es, die strategisch wichtigen 30 Prozent der Unternehmensressourcen zu steuern.[31] Hieraus ist zwar noch keine analytische Ableitung der konkreten Budgetwerte möglich, aber es wird eine Orientierungshilfe für den zielgerichteten Ressourceneinsatz zur Verfügung gestellt.

Beide Ansätze (Balanced Scorecard und Budgetierung) folgen einer ähnlichen Logik. Das Ergebnis ist, dass die strategischen Ressourcen, die nicht in Projekten bzw. strategischen Aktionen gebunden sind und somit laut Horvath und Partner etwa 20 Prozent des Budgets ausmachen, aus den strategischen Zielen abgeleitet werden, und dass somit das Verständnis für den Zusammenhang zwischen Budget und Strategie gefördert wird. Die aus den strategischen Zielen abgeleiteten benötigten Ressourcen sind dann die Detailbudgets, die es ausführlich zu planen gilt. Für die restlichen 70 Prozent der Ressourcen sind Globalbudgets ausreichend.[32]

Dennoch kann die Balanced Scorecard kein Ersatz für die Budgetierung sein, da sie weder erklärt, wie die benötigten Ressourcen zu bestimmen sind, noch wie die benötigten Einsatzfaktoren in monetäre Parameter zu übertragen sind. Ihr Nutzen besteht darin, eine Verbindung zwischen strategischen Absichten und operativem Handeln im Tagesgeschäft herzustellen.

Eine Kaskadierung der Balanced Scorecard auf die Unternehmensbereiche fördert die Integration der Strategie in der Organisation noch weiter und unterstützt die zielgerichtete Ressourcenvergabe im Budgetierungsprozess. Zudem kann der Einsatz einer Balanced Scorecard die Kommunikation von strategischen Zielen bestärken und auch zur Motivation hinsichtlich der Erreichung strategischer Ziele beitragen. Somit stellt sie eine hervorragende Ergänzung zum Budget dar, indem sie bei richtigem Einsatz ein strategisches Bewusstsein schafft und somit den Aufbau einer strategiefokussierten Organisation ermöglicht.

3.3 Komplexitätsreduktion und –beherrschung der Budgetierung

3.3.1 Komplexitätsreduktion durch Entfeinerung der Budgetierung

In den nachstehenden Abschnitten sprechen wir über die (geschätzten) 70% des Budgetierungsaufwandes, die nicht strategisch relevant sind, aber dennoch wegen der Außenwirkung auf die Stakeholder des Unternehmens für die Steuerungsbedarfe des Managements von Bedeutung sind. Die Performance einer Unternehmung wird überwiegend kalenderzeitbezogen und hier - wegen der nach außen gerichteten Publizitätspflichten - maßgeblich finanziell beurteilt. Die Beherrschung der finanziellen Performance geschieht überwiegend über die Budgetierung in den Ausgestaltungsformen Planung, Forecasting und laufende Berichterstattung. Motiviert durch die zunehmende Integration des internen und externen Rechnungswesens und den technologischen Siegeszug mehrdimensionaler Datenbanken tendierten Manager und Controller in der Vergangenheit zu einer zunehmenden Verfeinerung der

[31] *Horváth & Partners* (Hrsg.), Beyond Budgeting umsetzen (B), 2004, S. 177.
[32] *Horváth & Partners* (Hrsg.), Beyond Budgeting umsetzen (B), 2004, S. 138.

Budgetierungsinhalte: Immer mehr Kostenarten, immer mehr Kostenstellen; mehrdimensionale Erlösdarstellungen flossen in die Budgetierung ein und führten - besonders in der Planungsphase - zu einer Steigerung der Komplexität der Budgetierung, die allgemein als unangemessen beurteilt wird.

Die Frage, welche Feinheit (Granularität) ein Planungssystem oder gar allgemein ein Informationssystem haben sollte, wurde theoretisch früh gestellt, aber bis heute nicht befriedigend beantwortet. Die Ursache ist das ungelöste Problem der Nutzenermittlung von Informationen. Der Nutzen einer Information bestimmt wesentlich die Kosten, die mit ihrer Erhebung und Verfügbarmachung einhergehen dürfen. Es liegt nahe zu vermuten, dass der Zusammenhang prinzipiell einen Verlauf hat, wie er in Abb.14 dargestellt ist: die Steuerbarkeit eines Bereiches wächst zunächst stark mit der ersten, noch wenig detaillierten Information über führungsrelevante Sachverhalte. Mit zunehmender Verfeinerung der angebotenen Information jenseits eines Optimalpunktes sinkt der Informationsnutzen, weil Widersprüchlichkeiten entstehen können, der mentale Verarbeitungsaufwand für die Informationsaufnahme aber steigt usw.. Man kann weiter annehmen, dass der Nutzen aus Informationen und die Detaillierung von Information Pareto-verteilt sind. Das bedeutet: es steht zu erwarten, dass der Optimalpunkt der Informationsversorgung eher weit links auf der Abszisse liegt, also 20% der Informationen komplexer Budgetierungssysteme schon den Optimalpunkt der Feinheit darstellen. Konkret bedeutet dies: nicht 20 Kostenarten, sondern 4 Kostenartengruppen sollten budgetiert werden, oder: statt 50 Kostenstellen 5-10 Kostenstellenbereiche.

Mit der Entfeinerung der Budgetinhalte kann eine erhebliche Reduktion des Budgetierungsaufwandes erreicht werden. Es entstehen weniger Datenfriedhöfe in zentralen Abteilungen, die viel zu selten zu Rückfragen bei dezentralen Controllingabteilungen führen, dass man unterstellen kann, sie würden durch die Mitarbeiter in den Zentralen effizient analysiert.

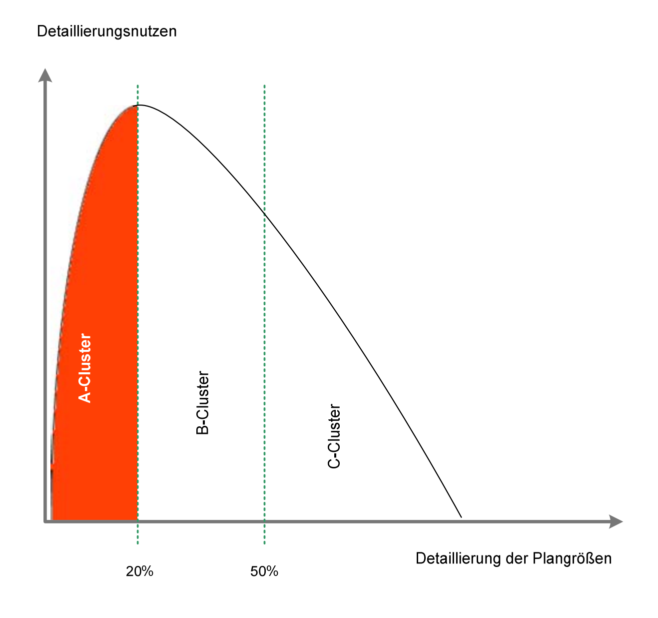


Abb. 14: Detaillierung von Budgetinhalten

3.3.2 Komplexitätsbeherrschung durch technologische Budgettierungsfunktionen

Parametrische Verteilung von Planaggregaten

In einigen Bereichen der Unternehmensplanung führt eine Reduktion der Planungsinhalte zu einem echten Informationsverlust. Dies ist besonders im Absatz-Umsatz und Deckungsbeitragsbereich zu sehen, in dem die Marktdynamik sich auch in einer zunehmenden Segmentierung der Märkte äußert. Die Mehrdimensionalität dieses Budgetierungsbereiches führt zu einem exponentiell steigenden Planungsaufwand: wenn man 100 Produkte an 1000 Kunden verkauft, dann führt die eindimensionale (Jahres-) Absatzeingabe der zu verkaufenden Produkte zu 100 Systemeingaben, die zweidimensionale (Jahres-) Absatzeingabe der zu verkaufenden Produkte und der zu bedienenden Kunden zu 100x1000 Systemeingaben (unterstellt, alle Kunden kaufen alle Produkte).

Intelligente Budgetierungssysteme müssen hier die Funktionalität aufweisen, ausgehend von der mehrdimensionalen Darstellung von Referenzdaten (historische Istdaten oder Benchmarks von Peers), durch die parametrische Angabe von (absoluten oder relativen) Veränderungen auf jeder Ebene des mehrdimensionalen Datenraumes eine rechnerische Verteilung der aggregierten Referenzgrößen auf die granularen Budgetwerte vorzunehmen.

Verteilung von Jahreswerten auf Monate mit Saisonverläufen

Eine Sonderanwendung dieser Funktion ist die Aufteilung von Jahreswerten des Budgets auf Monatswerte unter Nutzung von Saisonalverläufen. Ohne diese Funktion würde die Aufteilung des Jahresbudgets der obigen 100x1000 Systemeingaben auf 100x1000x12 steigen (unterstellt, alle Kunden kaufen alle Produkte in allen Monaten).

3.3.2 Komplexitätsreduktion durch Metadaten-Management

In der Budgetierung komplexer Unternehmen gibt es zahlreiche Metadaten. Metadaten sind „Informationen über Informationen“. Wenn Sie eine Exceldatei erhalten, die den Wert 500.000 enthält und den Namen „Umsatz122005kumIstFilialeAEur.xls“ trägt, dann sind die Metadaten im Dateinamen enthalten und lauten „der in Euro ausgedrückte kumulierte Umsatz per Dezember 2005 der Filiale A“. Die Excel-Tabelle hat dann nur noch den eigentlichen Wert, nämlich das Datum zum Inhalt.

Metadaten haben in einer Welt, in der unstrukturierte Informationen als Spreadsheets oder Webseiten allgegenwärtig sind, eine hohe Bedeutung bekommen. Selbst Spreadsheets, die als wohlgeordnete Tabellen daherkommen, sind unstrukturiert, weil der Zellinhalt meist beliebig ist und die beschreibenden Nachbarzellen ohne begriffliche Strenge sind und deshalb nur schwer in komplexe Informationssysteme integriert werden können Durch die weite Verbreitung von mehrdimensionalen Datenbanken, die in der Regel ein Excel-Frontend haben, wird die in Tabellenkalkulationen verloren gegangene Trennung von Daten und Metadaten wieder eingeführt. Es steht zu erwarten, dass die Verbreitung der Kenntnis von Metadaten mit der Diffusion von XML einhergeht: werden Spreadsheets als XML-Dateien erstellt

und verbreitet, dann sind die Metadaten strikt von den Daten getrennt. Wenn man - wie dies durch internationale Wirtschaftsprüferorganisationen bereits geschieht - die Metadaten dann für bestimmte Berichtsformate, beispielsweise den Jahresabschluss nach IAS/IFRS, standardisiert[33], dann spricht man von Taxonomien. Die taxonomische Publikation von Abschlussinformationen macht diese (beispielsweise für finanzielle Benchmarks) allen anderen Teilnehmern am Internet verfügbar.
Intern sollten die Metadaten für Unternehmen als Taxonomien verbindlich publiziert werden. Man erhält hierdurch eine weite Freiheit in der Wahl der anzuwendenden Softwareprogramme, weil XML-basierte Taxonomien von allen Anbietern von Budgetierungsprogrammen gelesen werden können. Man kann intern folgende Metadaten unterscheiden:

- **Obligatorische Taxonomien**: Obligatorische Taxonomien sind einzuführen für alle finanziellen Informationen (Konten) und deren Aggregationen oder Umformungen in betriebswirtschaftlichen Modellen (z.B. Free Cash-Flow-Definitionen), sowie für Zeitdimensionen und Dimensionen, die Metadaten für die Datenarten (Plan, Ist, Forecast,…) von Budgetinformationen beschreiben. Weiterhin empfiehlt sich, die Dimensionen, die die rechtliche und unternehmerische Führungsorganisation von Unternehmen beschreiben, verbindlich vorzuhalten.
- **Fakultative Taxonomien:** Diese Metadaten sind nicht zwingend in Budgetierungssystemen vorhanden. Beispiele sind: Produktinformationen, Kundeninformationen, Markt- oder Regionsbezeichnungen, Projektbezeichnungen, Prozessstrukturen.

3.3.4 Parametrische Treibermodelle

Budgetierungsarbeiten können durch parametrische Treibermodelle erheblich verkürzt, für weite Teilbereiche sogar automatisiert werden. Das Ergebnis des Budgetierungsprozesses stellt der Masterplan dar, bestehend aus dem Leistungs-, dem Finanz- und dem Bilanzbudget. Für Budgetierungssysteme kommen zwei unterschiedliche Modellierungsansätze in Frage:

- **Finanzorientierte Modellierung:** Aufbauend auf Konten, Kontenplänen oder Kennzahlen wird eine Erklärlogik für den Masterplan aufgebaut, die den Konventionen des Rechnungs- und Finanzwesens entspricht. Die Modelllogiken zielen auf die Gliederung und Festlegungen der finanziellen Führung ab. Die dazugehörigen Istdaten sind hierzu leicht zu ermitteln, die Gliederungslogik gibt den Managern aber nur abstrakte, vom Geschäftssystem des Unternehmens unabhängige Informationen zur Hand, die für ihn nur selten unmittelbar in Maßnahmen umsetzbar sind. So wird der Umsatz beispielsweise als Aggregation verschiedener Umsatzsteuer-Kategorien gezeigt.
- **Geschäftsorientierte Modellierung**: Bei dieser Modellierung werden die Treiber der finanziellen Budgetinhalte ermittelt und arithmetisiert. Die

[33] Siehe www.xbrl.org, wo man für zahlreiche betriebswirtschaftliche Inhalte Taxonomien findet, beispielsweise auch eine für das Performance-Management

Budgetwerte sind die abhängigen Variablen, die Treiber die unabhängigen Variablen. Dazu sind Rechenregeln und Parameter anzugeben. Der Umsatz als Budgetwert wird bei einem Dienstleistungsunternehmen deshalb hier als Anzahl fakturierbarer Leistungstage x durchschnittlicher Honorarsatz dargestellt und bei einem Serienhersteller als Produkt aus Nettoerlös der Serienprodukte und Absatzmenge.

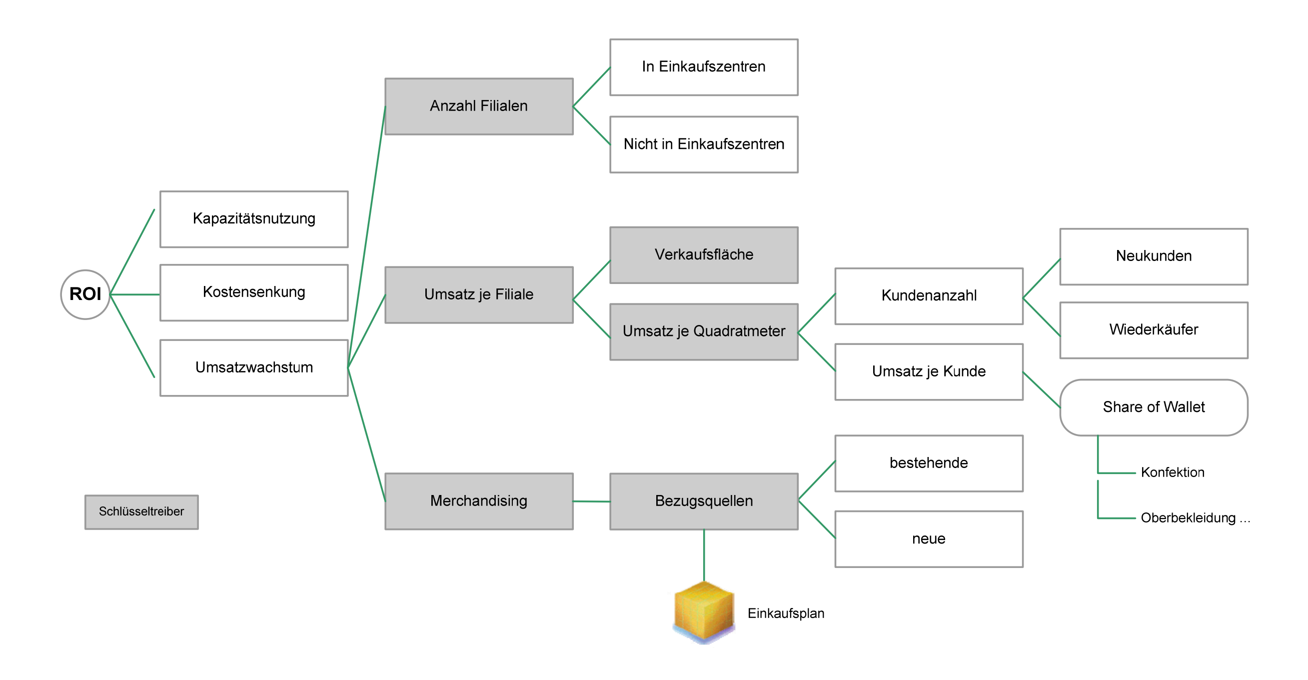


Abb. 15: Geschäftsorientierte Modellierung

Die Möglichkeit zur Effizienzsteigerung von Budgetierungssystemen ergibt sich aus der konsequenten Anwendung der finanz- und geschäftsorientierten Treibermodellierung. Wenn Treibermodelle für ein Unternehmen bestehen, dann kann der Aufwand für die Budgetierung auf die Ermittlung weniger Mengen- oder Parametereingaben begrenzt werden. Wenn in dem obigen Beispiel (Abb.15) die Anzahl der Filialen (ggfs. differenziert nach Größenklassen) und der parametrisch bestimmte Filialumsatz (je Größenklasse) feststeht, dann müssen nicht 100 Filialen bottom-up budgetiert werden, sondern man kann über Simulationen Rahmenparameter (Preissteigerungen,…) eingeben und eine qualitativ ausreichende Budgetwertfindung top-down oder kollaborativ ermitteln.

3.3.5 Prozessgebundene Treibermodelle

Ein Sonderfall von den dargestellten Treibermodellen stellen prozessgebundene Treibermodelle dar. Bei ihnen ist das Treibermodell im Prinzip die für ein bestimmtes Prozessergebnis erforderlichen Teilprozesse. Abb.16 zeigt, dass in Standardleistungsprofilen für die verschiedenen Leistungsprozesse jeweils aufgelistet ist, welche Kapazitätsbeanspruchung mit der Abwicklung eines Teilprozesses in den Leistungsstellen verbunden ist.

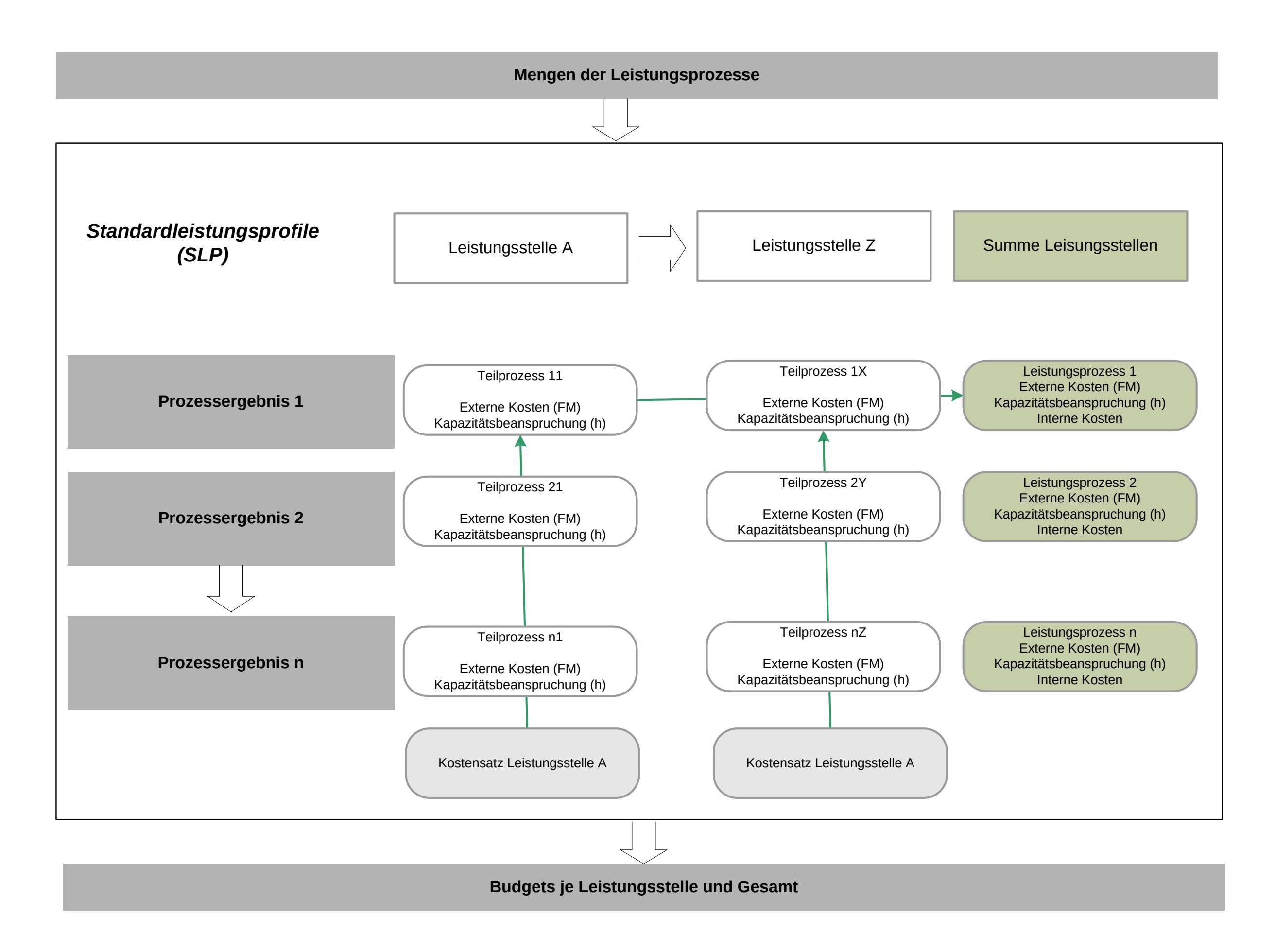


Abb. 16: Geschäftsorientierte Modellierung

Multipliziert mit den jeweiligen Kostensätzen der Leistungsstellen und ergänzt um die externen Kosten (z.B. Fertigungsmaterial) ergeben sich die aggregierten Kosten der Leistungsprozesse. Es wird deutlich, dass die Budgetierung durch die Eingabe der Leistungsprozessmengen abgearbeitet ist und als Ergebnis Kostenbudgets je Leistungsstelle ausgewiesen werden können. Dieses Basismodell kann natürlich noch um weitere parametrische Angaben, beispielsweise die Eingabe von Rationalisierungsvorgaben je Leistungsstelle, verfeinert werden.
Solche Treibermodelle können immer dann eingesetzt werden, wenn die Abarbeitung von Leistungsprozessen für den Leistungsbereich kennzeichnend ist. Beispiele hierfür sind Dienstleistungsunternehmen (Krankenhäuser, Banken, Versicherer) oder dienstleistende Unternehmensbereiche (IT).

3.3.6 Komplexitätsbeherrschung durch zeitliche Entflechtung der Budgetierung

Letztlich sei darauf hingewiesen, dass die Budgetierung in der zeitlichen Abfolge optimiert werden kann. Arbeitsintensiv wird eine Budgetierung immer dann, wenn interne Bereichsabstimmungen (zum Beispiel bei gegenseitigen Leistungsverflechtungen) erfolgen müssen. In diesen Fällen kann eine vorgezogene Publikation der Verrechnungspreise und laufende Abstimmungen der

Planungsprämissen zu einer zeitlich entspannten Kumulation der Budgetierung führen.

4 Best Practice in der Budgetierung

Strategie und Budgetierung sind ein weites und wichtiges Feld der Unternehmenssteuerung, Statt einer Zusammenfassung sollen in zwei Tabellen die hier dargelegten und von der Unternehmensempirie bestätigten Schlaglichter, getrennt nach der operativen Planung (Abb. 18) und der Budgetierung (Abb.19) noch einmal zusammengefasst werden:

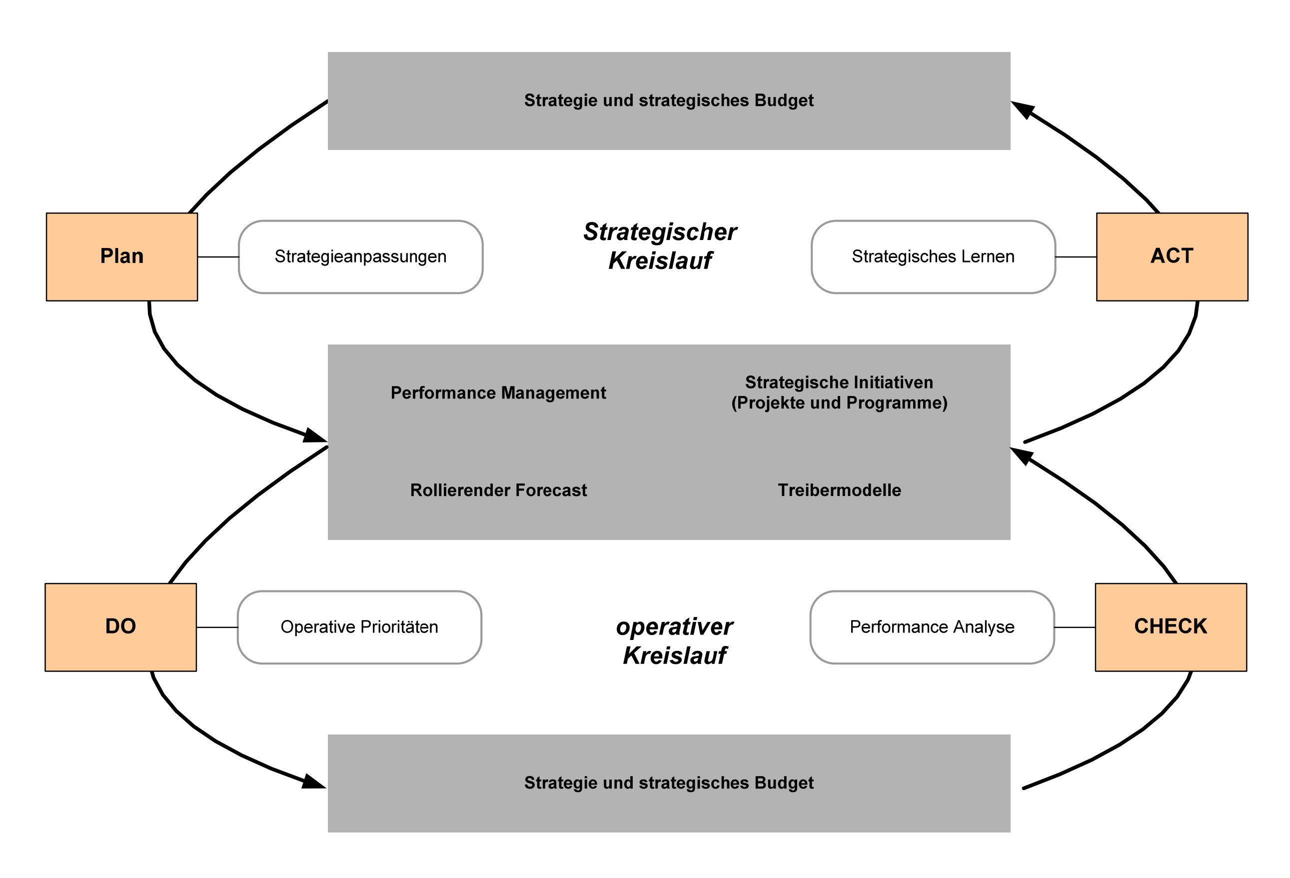


Abb. 17: Double-loop Budgeting

Die Summe aller hier dargestellten Weiterentwicklungen der Budgetierung zeigt eine Entwicklungsrichtung für dieses wichtige Managementinstrument vor, in der ihre Kernfunktionen der Koordination der strategischen Unternehmensaktivitäten und der flexiblen Anpassungen an wechselnde operative Performancelagen erhalten bleiben. Der Managementprozess der strategieumsetzenden Steuerung bekommt gewissermaßen zwei gleichberechtigte Kreisläufe (Abb. 17):

- In einem **strategischen Kreislauf** erfolgt die permanente Überprüfung und Anpassung der Unternehmensstrategie, die formalzielorientiert in einem strategischen Budget zusammengefasst wird.

- In einem **operativen Kreislauf** erfolgt eine kalenderzeitorientierte Überprüfung der Unternehmensleistung mit entsprechenden steuernden Eingriffen und Anpassungen der operativen Prioritäten.

Integration der Planungsebenen	Sachzielpläne sollen über Formalzielpläne integriert werden.
Klare Ziele und Zielniveaus setzen	Externe und interne Zielerwartungen über Wertorientierung integrieren. Relative Zielniveaus setzen. Ziele über Ursache-Wirkungsketten auf nicht-finanzielle ausdehnen und die Organisation kaskadieren.
Zielverantwortlichkeit etablieren	Dies wird erreicht durch: Fähigkeiten – Ressourcen – Kompetenzen – Zielvereinbarung – Anreize und Konsequenzen
Angemessenen Planungshorizont festlegen	Planungshorizonte werden bestimmt durch: - Den normalen Produktentwicklungs- und Verkaufszyklus einer Branche. - Die Vorlaufzeit für Entscheidungen über wesentliche Vorleistungen (Investitionen). - Die vom Management, Aufsichtsrat und anderen Stakeholdern als angemessen angesehene Zykluszeit.
Eine gemeinsame Sprache sprechen	Definitorische Klarheit schaffen. Standardprozesse der Planung einführen. Bereichsadäquate Formate der Planung und Berichterstattung einsetzen. Verbindliche Zeitliniensetzen und kontrollieren.
Taktiken (Maßnahmen, Projekte, Programme) treiben Budgets	Der häufigste Mangel der Budgetierung sind Budgets, die sich als Planungselement verselbstständigen. OP muss **taktikgetrieben** sein (-welche Maßnahmen, Projekte, Programme? -Welche Ergebnisse sollen sie erzielen? – Erreichen diese das Zielniveau?)
Risikobewusstes Planen	Die Risiken der Taktiken sind zu bewerten. Planungen sind oft akribisch bei den Dingen, die die geringsten Risiken haben und Risiko-unbewusst bei den strategisch wichtigen, aber riskanten Plänen: das sind in der Regel Projekte und Programme. Sie müssen die höchste Managementaufmerksamkeit in der Planung haben.
Projekte und Programme in die OP/FP integrieren	Projekte werden oft nach der projekteigenen Logik geplant und nicht in die FP des Unternehmens integriert. Klare Methoden der Steuerung des Projketlebenszyklus sind mit FP zu integrieren.

Abb. 18: Best Practice der operativen Planung

Budgets sind die finanziellen Ergebnisse von Taktiken	Teilpläne (TP) sind häufig Operative Budgets und Investitionspläne, sie sind eng mit der OP (Taktiken= dispositive operative Entscheide, Projekte, Programme) zu verbinden.
Starten Sie mit Mengen	Am besten mit Leistungsmengen arbeiten, hinter denen Standard-Leistungsprofile stehen.
Dezimieren Sie Details	Zu hohe Detaillierung hindert die Flexibilität der Planung: ein ungewünschter Nebenerffekt der „Genauigkeit" detaillierter Pläne ist, dass die Geschwindigkeit der Entscheidung durch sie stark herabgesetzt wird.
Konzentrieren Sie sich auf Materialität und Volatilität	Wenn die Kosten pro Verbrauchseinheit sinken, steigen oft die absoluten Kosten stark an (Beispiel: Handys). Also in der Effizienzsteuerung auf relevante Kostenarten (A-Kostenarten) und auf B-Kostenarten mit starker Steigerungstendenz konzentrieren. Von 100 Kostenarten sind dies in der Regel 20-30.
Individualisierte Budgets statt starrer Planungsschedulen	Kostentreiber können je Kostenstelle/Verantwortungsbereich stark unterschiedlich sein. Die Mengengerüste der Finanzplanung müssen dieser Individualität entsprechen, damit die Budgets auch anerkannt werden.
Bauen Sie Planungs-dynamiken ein	Lassen Sie Simulationen und What...If-Analysen in die Finanzplanungen einfließen und archivieren Sie diese Bausteine für spätere Entscheidungssituationen.
Eliminieren Sie Politik	Wenn die Politik von dem Planungsprozess Besitz nimmt, sind die Verbindungen von Finanzplänen mit dem strategischen Plan und dem wirtschaftlichen Nutzen nicht sichergestellt.
Bauen Sie Planungs-dynamiken ein	Dynamische Budgetierungsprozesse sind Best Practice, in denen aktuelle Informationen in die Budgetierung und die Forecasts einfließen. Alternative Planentscheide werden an mögliche Schwellenwerte für Ist-Entwicklungen gekoppelt und ausgelöst. Durch dynamische Budgetierung werden die kalenderorientierten Planungen flexibilisiert.

Abb. 19: Best Practice der Budgetierung

5 Literaturverzeichnis

Bücher

Bea, Franz Xaver; Haas, Jürgen [Grundwissen]: „Strategisches Management“ Stuttgart 2001.

Brühl, Rolf [Controlling (B), 2004]: Controlling – Grundlagen des Erfolgscontrollings, München; Wien: Oldenbourg Verlag, 2004

Buchner, Holger [Planung im turbulenten Umfeld (B), 2002]: Planung im turbulenten Umfeld - Konzeption idealtypischer Planungssysteme für Unternehmenskonfigurationen, München: Vahlen, 2002

Horváth, Péter [Controlling (B), 2003]: Controlling, 9., vollständig überarbeitete Auflage, München: Vahlen, 2003

Horváth & Partner (Hrsg.) [Balanced Scorecard umsetzen (B), 2001]: Balanced Scorecard umsetzen, 2., überarbeitete Auflage, Stuttgart: Schäffer Poeschel Verlag, 2001

Horváth & Partners (Hrsg.) [Beyond Budgeting umsetzen (B), 2004]: Beyond Budgeting umsetzen – Erfolgreich planen mit Advanced Budgeting, Stuttgart: Schäffer Poeschel Verlag, 2004

Egger, Anton/Winterheller, Manfred [Budgetierung (B), 2002]: Kurzfristige Unternehmensplanung: Budgetierung, 12., unveränderte Auflage, Wien: Linde, 2002

Grünig, Rudolf [Planung und Kontrolle (B), 2002]: Planung und Kontrolle: Ein Ansatz zur integrierten Erfüllung der beiden Führungsaufgaben, 3. Auflage, Bern; Stuttgart; Wien: Haupt, 2002

Kaplan, Robert S./Norton, David P. [Die strategiefokussierte Organisation (B), 2001]: Die strategiefokussierte Organisation – Führen mit der Balanced Scorecard, Stuttgart: Schäffer Poeschel Verlag, 2001

Peemöller, Volker [Grundlagen und Einsatzgebiete des Controlling, 1997]: Controlling: Grundlagen und Einsatzgebiete, Herne/ Berlin:1 997

Porter, Michael E. [Wettbewerbsstrategie (B), 1999]: Wettbewerbsstrategie (Competitive Strategy) – Methoden zur Analyse von Branchen und Konkurrenten, 10., durchgesehene und erweiterte Auflage, Frankfurt/Main; New York: Campus Verlag, 1999

Wall, Friederike [Planungs- und Kontrollsysteme (B), 1999]: Planungs- und Kontrollsysteme: informatorische Perspektive für das Controlling; Grundlagen – Instrumente – Konzepte, Wiesbaden: Gabler, 1999

Weber, Jürgen; Schäffer, Utz; Weber [Einführung 2008]: Einführung in das Controlling. 12., überarb. und aktualisierte Aufl. Stuttgart: Schäffer-Poeschel.

Welge, Martin K./ Al-Laham, Andreas [Management]: „Strategisches Management – Grundlage – Prozesse – Implementierung-“, Wiesbaden 2001.

Wild, Jürgen [Grundlagen der Unternehmensplanung (B), 1982]: Grundlagen der Unternehmensplanung, 4. Auflage, Opladen: Verlag für Sozialwissenschaften, 1982

Aufsätze

Amrein, Silvia/ Widmer, Markus/ Witmer, Dirk E. [Better Budgeting, 2003]: „Better Budgeting“ – Planungsansatz der UBS, in: Horváth, Péter (Hrsg.), Performancesteigerung und Kostenoptimierung – Neue Wege und erfolgreiche Praxislösungen, S. 267-294

Daum, Jürgen [Ein Management- und Controlling-Modell für nachhaltigen Unternehmenserfolg, 1999]: Beyond Budgeting: Ein Management- und Controlling-Modell für nachhaltigen Unternehmenserfolg, in: Klein, Andreas/ Vikas, Kurt/ Zehetner, Karl (Hrsg.), Der Controlling-Berater: Informationen, Instrumente, Praxisberichte, Band 1: Gruppenübersicht: Wegweiser, Controlling-Trends, Controlling-Werkzeuge, Lexikon, S. 2/397-2/429

Gaiser, Bernd/ Gleich, Ronald [Die Evolution ist machbar, 2004]: Budgetierung: Die Evolution ist machbar, in: FAZ Frankfurter Allgemeine Zeitung, 14. Juni 2004, S. 26

Gleich, Ronald/ Kopp, Jens [Ansätze zur Neugestaltung der Planung und Budgetierung, 2001]: Ansätze zur Neugestaltung der Planung und Budgetierung – methodische Innovationen und empirische Erkenntnisse, in: Controlling, Heft 08-09/ 2001, S. 429-436

Kaplan, Robert S./Norton, David P. [Strategiekarten (A), 2004]: Wie Sie die Geschäftsstrategie den Mitarbeitern verständlich machen, in: Harvard Business Manager - Edition, 01/2004, S.54 - 64

Pfläging, Niels [Fundamente des Beyond Budgeting, 2003]: Fundamente des Beyond Budgeting – Controller als Akteure bei der Realisierung eines integrierten Modells zur Unternehmenssteuerung, in: CM controller magazin, Heft 02/2003, S. 188-197

Porter, Michael. E. [What Is Strategy? (A), 1996]: What Is Strategy?, in: Harvard Business Review, Jg. 74, November/December 1996, S. 61 - 78.

Weber, Jürgen/ Lindner, Stefan, [(Better) Budgeting oder Beyond Budgeting (A), 2004]: (Better) Budgeting oder Beyond Budgeting? – Eine Analyse aus koordinationstheoretischer Perspektive, in: Controller Magazin 29 (2004), S. 224 - 228.

Methoden zur Untersuchung komplexer Datenmodelle

Dieter Riebesehl

Abstract: *Datenmodelle sind ein zentraler Bestandteil betrieblicher Informationssysteme. In der Praxis eingesetzte Datenmodelle erreichen oft eine erhebliche Komplexität. Es gibt verschiedene Ansätze, die Struktur von Datenmodellen zu untersuchen und wiederkehrende Muster zu entdecken. Eine Möglichkeit zur Reduktion der Komplexität besteht darin, gleiche oder ähnliche Teilstrukturen zu identifizieren. Durch Ausnutzung solcher Strukturanalogien kann dann das Datenmodell vereinheitlicht und vereinfacht werden: Auffinden von Relationen, die zusammengelegt werden können, Hinweise zur Denormalisierung, Identifikation neuer sinnvoller Relationen und Ansätze zur Entdeckung von Patterns.*

1 Einleitung

Unter Strukturanalogien zwischen Datenmodellen werden allgemein Ähnlichkeiten zwischen verschiedenen Teilen eines Modells verstanden. Es sind verschiedene Möglichkeiten vorstellbar, solche Analogien zu definieren. Diese werden für unterschiedliche Modellierungsmethoden verschieden ausfallen. Datenmodelle werden im Allgemeinen grafisch dargestellt, deshalb können Analogien in Form von isomorphen oder homomorphen Teilgraphen vorliegen. Man kann auch von den Informationen ausgehen, die über ein Datenmodell in einem *data dictionary* abgelegt werden, dann können Analogien zwischen Objekten des *data dictionary* vorliegen, also typischerweise zwischen Entitäten und/oder Relationen. Ein Vorschlag, Analogien zwischen Relationen durch eine Maßzahl zu charakterisieren, findet sich in [FE05]. Als Maß für die Ähnlichkeit zweier Relationen wird die normierte symmetrische Differenz der Entitätenmengen, auf denen die Relationen definiert sind, genommen.

In dieser Arbeit soll dieser Ansatz weiter verfolgt werden. Es werden zwei Modellierungsvarianten und die daraus resultierenden Unterschiede betrachtet. Für beide Modellierungsmethoden werden Ähnlichkeitsmaße definiert. Beispielhaft werden einige der Referenzdatenmodelle aus [SC97] mit den vorgestellten Methoden untersucht. Dazu werden passende Werkzeuge mit dem CAS-System Mathematica® entwickelt.

2 Ähnlichkeitswerte und Strukturanalogien

Grundsätzlich kann die Identifikation von Strukturanalogien auf zwei Wegen erfolgen:

1. Die grafische Darstellung der Datenmodells kann – per Augenschein oder durch graphentheoretische Methoden – auf Analogien hin untersucht werden, oder
2. aus einer abstrakten Repräsentation des Datenmodells – etwa im *data dictionary* – können mittels eines Algorithmus numerische Maßzahlen für das Vorliegen von Analogien gewonnen werden.

Hier soll hauptsächlich der zweite Weg verfolgt werden. Allerdings lässt sich zeigen, dass durch eine leichte Modifikation des verwendeten Algorithmus brauchbare Schritte in Richtung auf die erste Vorgehensweise getan werden können.

Es soll zunächst einmal die Definition des Ähnlichkeitswertes aus [FE05] wiederholt werden. Dazu sei ein Datenmodell mit Entitäten und Relationen gegeben, $\mathcal{R}$ sei die Menge der Relationen, und $\mathcal{E}$ die Menge der Entitäten. Die Behandlung von als Entitäten reinterpretierten Relationen sei zunächst zurückgestellt. Dann gibt es zu einer Relation $R \in \mathcal{R}$ die Menge $\hat{E}(R) \subseteq \mathcal{E}$ der Entitäten, über denen R definiert ist. $\hat{E}(R)$ ist i.A. eine Multimenge, in der Elemente mit Vielfachheit (mehrfach) vorkommen können.

Bezeichnet man für eine Multimenge $\hat{M}$ und ein Element $a \in \hat{M}$ mit $r_{\hat{M}}(a)$ die Vielfachheit von a in $\hat{M}$, dann kann man Durchschnitt und Vereinigung von Multimengen wie folgt definieren:

$$\begin{aligned} r_{\hat{M}\cap\hat{N}}(a) &:= \min(r_{\hat{M}}(a), r_{\hat{N}}(a)), \\ r_{\hat{M}\cup\hat{N}}(a) &:= \max(r_{\hat{M}}(a), r_{\hat{N}}(a)). \end{aligned}$$

Definiert man die Mächtigkeit für eine Multimenge durch

$$|\hat{M}| := \sum_{a\in\hat{M}} r_{\hat{M}}(a),$$

dann ist der Ähnlichkeitswert zweier Relationen einfach gegeben durch

$$a(R_1, R_2) = \frac{|\hat{E}(R_1) \cap \hat{E}(R_2)|}{|\hat{E}(R_1) \cup \hat{E}(R_2)|}.$$

Bei der Definition des Ähnlichkeitswertes ist natürlich die Definition von $\hat{E}(R)$ entscheidend, und diese wiederum hängt vom Typ des verwendeten Datenmodells ab. In den folgenden Abschnitten sollen zwei Datenmodelle näher betrachtet werden:

ERM: Das Standard-ERM nach Chen ([CH76]), welches nur Entitäten, dargestellt durch Rechtecke, und Relationen, dargestellt durch Rauten, unterscheidet und insbesondere keine als Entitäten reinterpretierten Relationen kennt.

PERM: Das erweiterte ERM nach Loos ([LO97]), welches reinterpretierte Relationen kennt. Es enthält noch weitere Modellierungsmöglichkeiten, auf die hier aber nicht eingegangen wird.

Die Verwendung der Abkürzung ERM für ein Modell ohne reinterpretierte Relationen entspricht nicht unbedingt dem Standard, wird aber von manchen Autoren von in die Modellierung einführender Literatur und auch in mehreren Modellierungswerkzeugen so genutzt, siehe z.B. Jarosch in [JA02].

2.1 Definition für verschiedene Modelltypen

In diesem Abschnitt sollen die Definitionen des Ähnlichkeitswertes für die beiden Modellierungsvarianten gegeben und diskutiert werden. Zur Veranschaulichung und zum Vergleich der Auswirkungen wird ein Ausschnitt aus dem Referenzdatenmodell zur Bedarfsplanung nach Scheer ([SC97], S. 176, Abb. B.I.66) in beiden Varianten dargestellt. Die Vorgehensweise bei der Umwandlung eines PER-Modells in ein ER-Modell ist wegen der unterschiedlichen Semantik nicht von vornherein klar, und es sind verschiedene Möglichkeiten denkbar. Da sich die Aufgabe einer solchen Umwandlung in der Praxis kaum stellt, soll hier keine vollständige Übergangsvorschrift gegeben werden. Einge Beispiele werden in Abschnitt 2.12 gegeben.

2.1.1 PERM

Da der Ähnlichkeitskoeffizient für das PERM entwickelt wurde, wird hier mit der Darstellung des Modellausschnitts im PERM begonnen, siehe Abbildung 1.

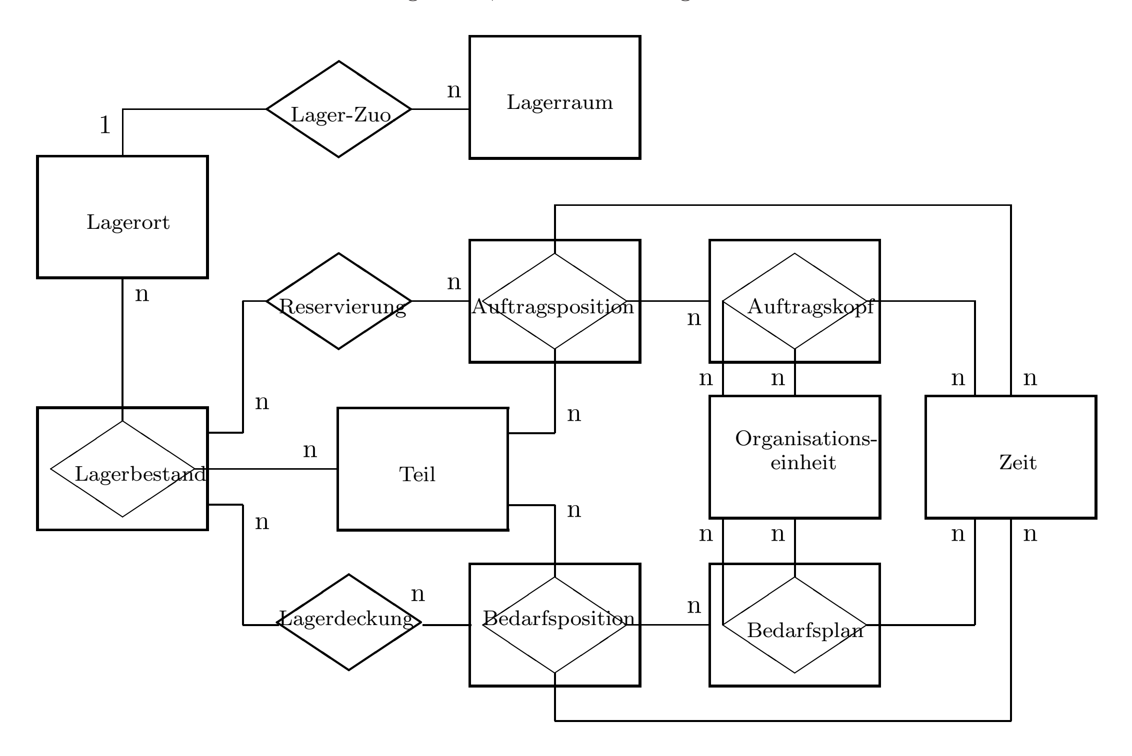


Abbildung 1: *Ausschnitt aus der Bedarfsplanung*

Definition von $\hat{E}(R)$

Die Mengen $\mathcal{E}$ und $\mathcal{R}$ haben hier einen nichtleeren Durchschnitt bestehend aus den Relationen, die als Entitäten reinterpretiert werden.

Die Bestimmung der Mengen $\hat{E}(R)$ geschieht rekursiv:

- Für $E \in \mathcal{E} \setminus \mathcal{R}$ ist

$$\hat{E}(E) := \{E\}.$$

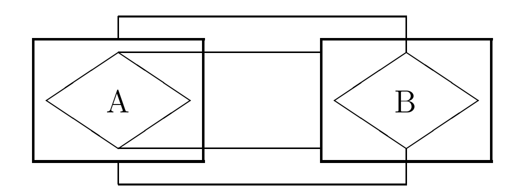
- Für $R = E_1 \times E_2 \times \cdots \times E_n \in \mathcal{R}$ ist

$$\hat{E}(R) := \bigoplus_{1}^{n} \hat{E}(E_i).$$

Dabei ist die Summe zweier Multimengen definiert durch

$$r_{\hat{M} \oplus \hat{N}}(a) := r_{\hat{M}}(a) + r_{\hat{N}}(a).$$

Damit die Rekursion endet, müssen rekursive Bezüge, wie sie in der nachfolgenden Abbildung zu sehen sind, explizit ausgeschlossen werden:



Aus den damit berechneten Ähnlichkeitskoeffizienten ergeben sich für den Modellausschnitt folgende Gruppen strukturidentischer Relationen, also solcher mit Ähnlichkeitswert 1:

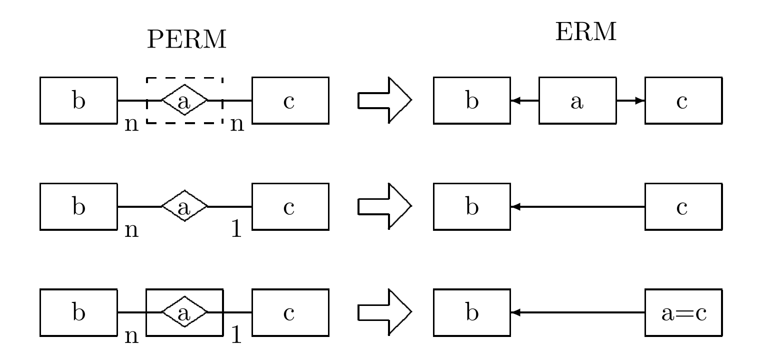
```
Auftragskopf = Bedarfsplan
Auftragsposition = Bedarfsposition
Lagerdeckung = Reservierung
```

Die Modellierung im PERM gestattet es, durch geschickte Anordnung der Objekte im Diagramm diese Analogien direkt über Symmetrien zu erkennen - so wie im obigen Bild.

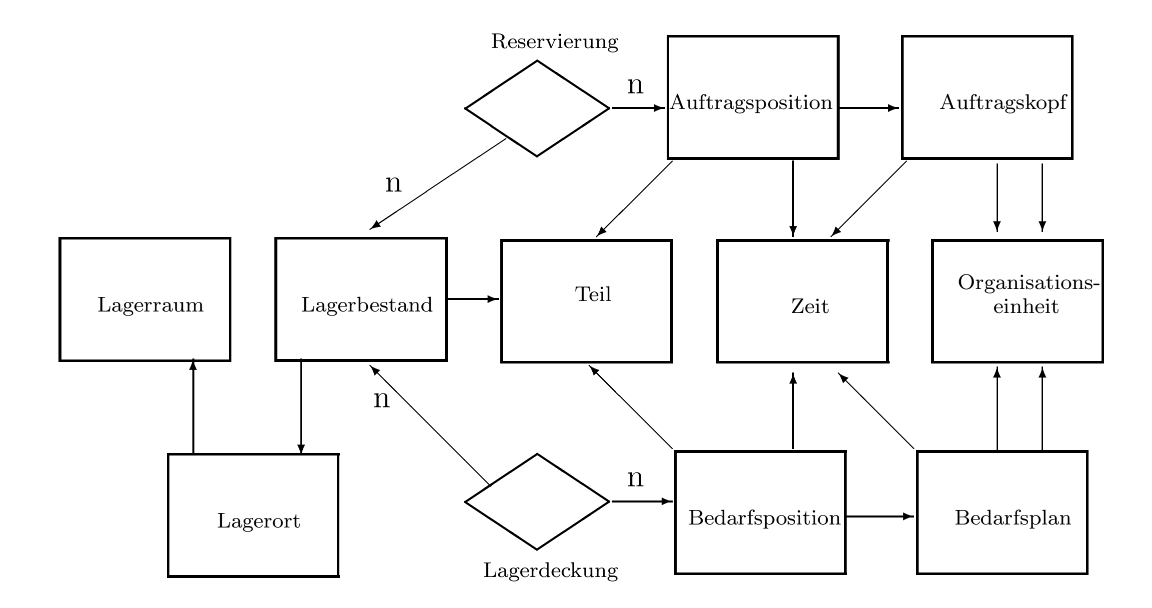
2.1.2 ERM

Da es im ER-Modell keine reinterpretierten Relationen gibt, sind Relationen stets Aggregationen aus Entitäten. Außerdem können zweistellige Relationen der Kardinalität $1 : n$ im Diagramm weggelassen werden – so wie ihnen auch keine eigenen physischen Tabellen entsprechen –, da sie durch Fremdschlüssel in den anderen Tabellen abgebildet werden. In den folgenden Diagrammen sind sie daher durch Pfeile dargestellt. Der Pfeil zeigt dabei von der Entität mit dem Fremdschlüssel auf die referenzierte Entität, also in Richtung auf die 1 in der Kardinalität.

Die Umwandlung eines PERM in ein ERM wird wie in der nächsten Abbildung dargestellt vorgenommen:



Der Modellausschnitt stellt sich im ERM nun so dar:

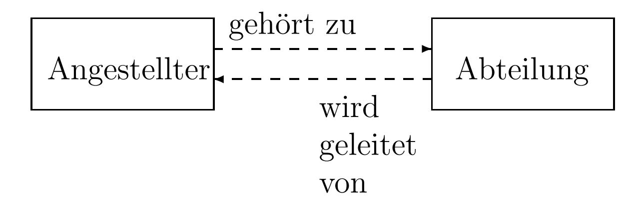


Die beiden gezeigten Relationen der Kardinalität $n : n$ könnten auch noch durch Entitäten ersetzt werden, dabei entstehen je zwei neue 1:n-Beziehungen. Strukturähnlichkeiten lassen sich hier an der Struktur des Graphen erkennen, der symmetrische Teile enthält. Dazu bedarf es jedoch einer geschickten Anordnung der Entitäten, die bei komplexen Modellen nur schwierig zu erreichen ist.

Der in [FE05] definierte Ähnlichkeitswert ist zwar formal auch im ERM definierbar, führt aber zu wenig interessanten Ergebnissen, da er nur feststellen kann, ob zwei Relationen direkt auf denselben Entitäten definiert sind. Weitergehende Ähnlichkeiten können nicht entdeckt werden: im obigen Bild wird die Strukturidentität zwischen `Lagerdeckung` und `Reservierung` nicht erkannt. Es gibt außerdem überhaupt nur diese beiden Relationen, die verglichen werden können.

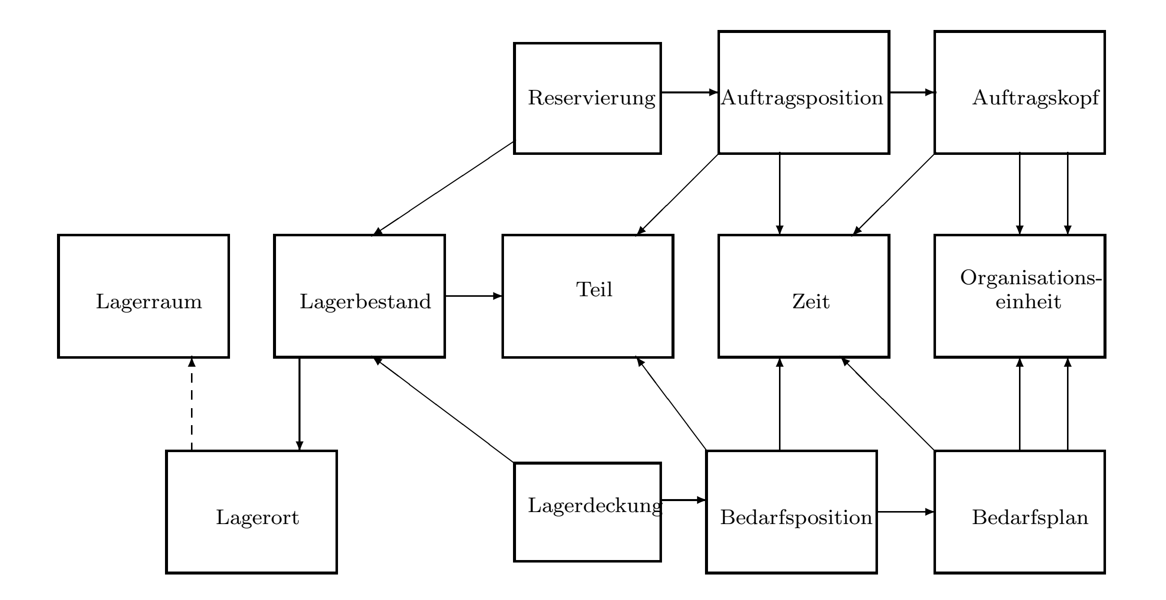
Eine Lösungsmöglichkeit für dieses Problem besteht darin, ganz auf Relationen als Modellierungselement zu verzichten und stattdessen alle Relationen durch Entitätstypen zu ersetzen - die sogenannten Koppelentitäten (vereinfachtes oder physisches ERM). Beziehungen zwischen Entitäten werden durch Fremdschlüssel in den zugehörigen Tabellen dargestellt, die Kardinalität der Beziehungen ist durchgehend 1:n und wird durch einen Pfeil dargestellt. Ähnlichkeitswerte werden nun zwischen Entitätstypen berechnet.

Dazu wird zunächst jedem Entitätstyp E die Multimenge $\hat{E}(E)$ von Entitätstypen zugeordnet, von der er über Beziehungen abhängig ist. Diese Zuordnung geschieht wieder rekursiv, Details siehe weiter unten. Dabei besteht die Gefahr von Zyklen, wie das Beispiel



zeigt. Hier sind `Angestellter` und `Abteilung` zyklisch voneinander abhängig. Solche Zyklen lassen sich vermeiden, wenn man unterscheidet, ob ein Fremdschlüssel Teil des Primärschlüssels ist oder nicht, d.h. ob die zugehörige Beziehung identifizierend ist oder nicht. Diese wichtige Unterscheidung wird von den meisten Autoren für das ER-Modell auch gemacht. Nicht-identifizierende Beziehungen werden dann gestrichelt dargestellt, wie im Bild oben.

Das Bild auf der vorigen Seite sieht mit diesen Änderungen so aus:



Die Definition von $\hat{E}(E)$ geschieht nun ganz pragmatisch über die Schlüssel – Primär- wie Fremdschlüssel – die in der Tabelle für den Entitätstyp E vorkommen. Dazu werden ein paar vereinfachende Annahmen getroffen, die die Allgemeinheit des Modells nicht einschränken[1]:

1. Entitätstypen E, von denen keine identifizierenden Beziehungen ausgehen, haben als Schlüssel ein einziges Attribut id_E und keine Fremdschlüssel. Es gibt also für diese keine zusammengesetzten Primärschlüssel. Diese Entitätstypen heißen *unabhängig*. Ein solcher Schlüssel heißt *freier Schlüssel*.

2. Entitätstypen E, von denen identifizierende Beziehungen $E \rightarrow E'$ ausgehen, übernehmen den Primärschlüssel $id_{E'}$ von jedem E' als Primärschlüsselteil. Ihr Primärschlüssel ist also im allgemeinen zusammengesetzt. Sie dürfen außerdem noch höchstens ein weiteres freies Primärschlüsselattribut id_E besitzen, welches nicht von einer identifizierenden Beziehung herrührt.

3. Für jede nichtidentifizierende Beziehung $E \rightarrow E'$ wird der Primärschlüssel $id_{E'}$ als Fremdschlüssel in E übernommen.

Man beachte, dass die übernommenen Primärschlüssel ihrerseits zusammengesetzt sein können. Im Ergebnis enthält dann die Tabelle für einen Entitätstypen eine Multimenge von freien Primärschlüsseln. $\hat{E}(E)$ ist dann einfach die Multimenge der zugehörigen Entitätstypen.

Wenn man annimmt, dass nur unabhängige Entitätstypen freie Schlüssel haben, dann kann man die Mengen $\hat{E}(E)$ direkt der graphischen Darstellung entnehmen. Alternativ kann man zur Bestimmung von $\hat{E}$ auf die DDL zurückgreifen, die die Tabellen definiert.

Im Vergleich mit der Darstellung des Modells im PERM ist die Relation `Lager_Zuo` verschwunden. Sie ist als Fremdschlüssel in der Entität `Lagerort` enthalten. Für die Ähnlichkeitswerte mit `Lagerbestand` erhält man im PERM:

$$\hat{E}(\text{ Lager_Zuo }) = \{ \text{ Lagerraum , Lagerort } \},$$

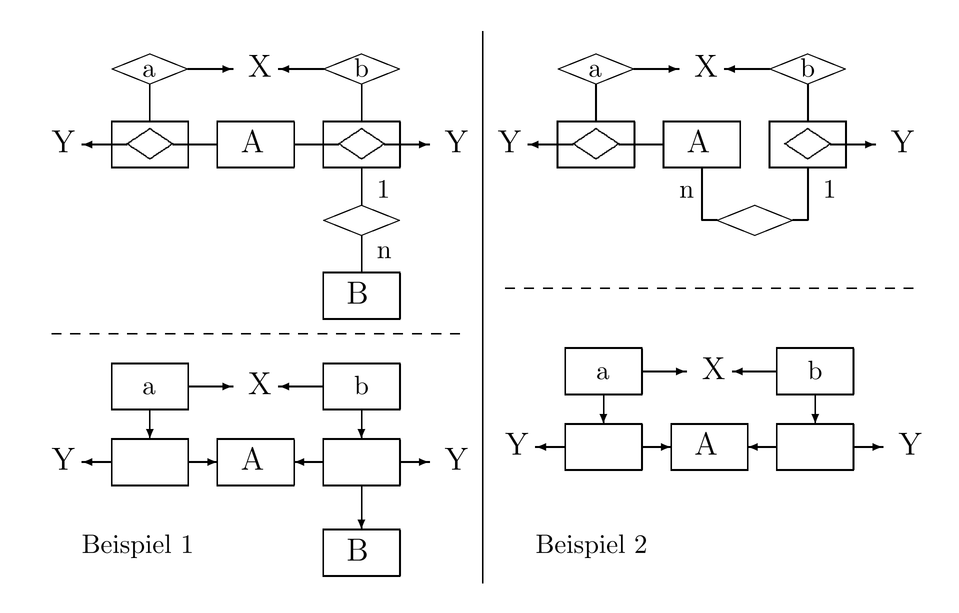
[1] Vorsicht: manche Tools zur Datenmodellierungs lassen zyklische Abhängigkeiten über identifizierende Beziehungen zu, ohne dabei zu bemerken, dass sie sich einen unendlichen Regress von Fremdschlüsseln einhandeln!

$$\hat{E}(\text{ Lagerbestand }) = \{\text{ Teil , Lagerort }\},$$
$$a(\text{ Lager_Zuo , Lagerbestand }) = \frac{1}{3}.$$

Im ERM werden stattdessen `Lagerort` und `Lagerbestand` verglichen:

$$\hat{R}(\text{ Lagerort }) = \{\text{ Lagerraum , Lagerort }\},$$
$$\hat{R}(\text{ Lagerbestand }) = \{\text{ Teil , Lagerort }\},$$
$$a(\text{ Lagerort , Lagerbestand }) = \frac{1}{3}.$$

Man gewinnt also hier denselben Ähnlichkeitswert. Das ist aber die Ausnahme. Meist ändern sich die Ähnlichkeitswerte. Dazu soll weiter unten in Abschnitt 3.2 ein detaillierteres Beispiel vorgestellt werden. Es können sogar Strukturidentitäten verschwinden oder neu entstehen:



In den beiden Beispielen ist ein fiktiver Datenmodellauschnitt oben als PERM, unten als ERM dargestellt. X und Y stehen für nicht näher ausgeführte, aber identische Modellteile.

Beispiel 1: Im PERM sind die Relationen a und b strukturidentisch,

$$\hat{E}_{\mathrm{PERM}}(a) = \hat{E}_{\mathrm{PERM}}(b) = \{A, \ \ldots\}.$$

Die Punkte stehen hier und im Folgenden für die Entitätenmengen, die aus X und Y herkommen.

Im ERM hingegen gewinnt b über die aufgelöste 1:n-Relation einen Schlüssel hinzu:

$$\hat{E}_{\mathrm{ERM}}(a) \neq \hat{E}_{\mathrm{ERM}}(b) = \{A, B, \ \ldots\}.$$

Beispiel 2: Im PERM sind die Relationen a und b *nicht* strukturidentisch,

$$\hat{E}_{\mathrm{PERM}}(a) = \{A, \ \ldots\} \neq \hat{E}_{\mathrm{ERM}}(b).$$

Im ERM hingegen gewinnt b über die aufgelöste 1:n-Relation den Schlüssel von A hinzu:

$$\hat{E}_{\mathrm{ERM}}(a) = \hat{E}_{\mathrm{ERM}}(b) = \{A, \ \ldots\}.$$

Der Ähnlichkeitswert für das ERM ist besonders aussagekräftig, da er unmittelbar ein Maß für die Ähnlichkeit der Schlüssel darstellt. Er unterscheidet auch identifizierende von nicht-identifizierenden Relationen und behandelt 1:n- und n:n-Relationen unterschiedlich. Strukturidentische Entitätstypen können ohne Weiteres in einer Tabelle zusammengefasst werden, da die Schlüssel identisch sind.

Ist der Ähnlichkeitswert kleiner als 1, so ist es manchmal möglich, die zu einem neuen Entitätstypen zusammengelegten Entitäten als Denormalisierung zu deuten, so im obigen Beispiel: fasst man `Lagerort` und `Lagerbestand` in einer Tabelle zusammen, so hat diese als Schlüssel

$$(\underline{\mathrm{id}}_{Teil}, \underline{\mathrm{id}}_{Lagerort}, \mathrm{id}_{Lagerraum}),$$

das ist eine denormalisierte Version der beiden Entitätstypen, bei der die zweite Normalform nicht erfüllt ist. Obwohl der Ähnlichkeitswert gering ist, zeigt dies keine besonders geringe Qualität der Denormalisierung an. Die Brauchbarkeit der Denormalisierung ist im Gegenteil eher schlechter, wenn die Menge der gemeinsamen Schlüssel der beteiligten Entitäten und damit auch der Ähnlichkeitswert groß ist. Beispielsweise haben `Bedarfsposition` und `Lagerdeckung` den Ähnlichkeitswert $\frac{5}{7}$, die Zusammenfassung der beiden in einer Tabelle führt zum Schlüssel

$$(\underline{\mathrm{id}}_{Lo}, \underline{\mathrm{id}}_{T}, \underline{\mathrm{id}}_{T}, \underline{\mathrm{id}}_{Z}, \underline{\mathrm{id}}_{Z}, \underline{\mathrm{id}}_{O}, \underline{\mathrm{id}}_{O}),$$

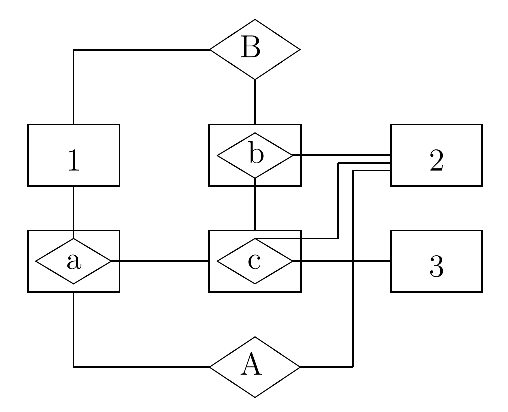
mit den Abkürzungen Lo, T, Z, O für Lagerort, Teil, Zeit und Organisationseinheit. Die Abhängigkeiten zwischen Nichtschlüsselattributen und Schlüsseln sind nun kaum mehr nachvollziehbar.

2.2 Ein alternatives Ähnlichkeitsmaß

In [FE05] wird vorgeschlagen, strukturidentische Entitäten zu einer Entität zu vereinigen. In den bisher betrachteten Beispielen ist dies sinnvoll gewesen und führte zu einer Vereinfachung des Datenmodells. Es ist aber ganz leicht, Beispiele zu konstruieren, in denen eine solche Identifikation problematisch ist.

2.2.1 Probleme bei der Nutzung von Strukturidentitäten

Man betrachte das folgende formale Datenmodell als Beispiel:



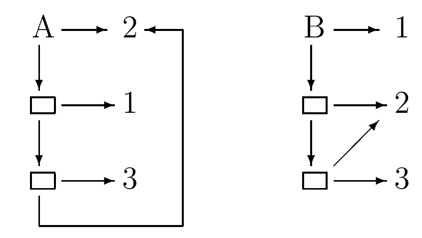
Man rechnet sofort $a(A, B) = 1$ nach. Dennoch lässt sich das Diagramm nicht sinnvoll dadurch vereinfachen, dass man A und B zu einer Relation zusammenlegt. Es gibt Strukturunterschiede, die sich in verschiedener Weise manifestieren:

1. Zwar „erben“ A und B die gleichen Fremdschlüssel in gleicher Anzahl von den drei Entitäten 1,2 und 3, aber sie sind jeweils anders „geschachtelt“:

$$\begin{aligned} A &= A(\mathrm{id}_2, [\mathrm{id}_1, [\mathrm{id}_2, \mathrm{id}_3]]) \\ B &= B(\mathrm{id}_1, [\mathrm{id}_2, [\mathrm{id}_2, \mathrm{id}_3]]) \end{aligned}$$

   Mit dieser geschachtelten Liste von Fremdschlüsseln oder Entitäten hat man zugleich einen neuen Kandidaten für ein besseres Ähnlichkeitsmaß.

2. Man kann den Relationen A und B Subgraphen des Datenmodells zuordnen:



   Diese sind nicht isomorph, auch nicht als ungerichtete Graphen, da sie u.a. verschieden lange Zyklen enthalten.

   Die Isomorphie dieser gerichteten Subgraphen könnte ein weiteres neues Kriterium für Strukturidentität sein.

2.2.2 Ein strukturiertes Ähnlichkeitsmaß

Eine geschachtelte Liste von Entitäten wie im obigen Beispiel kann leicht als neues Ähnlichkeitsmaß formal definiert werden. Zunächst wird jeder Relation R eine Struktur $E_s(R)$ zugeordnet, die rekursiv definiert wird (für PERM):

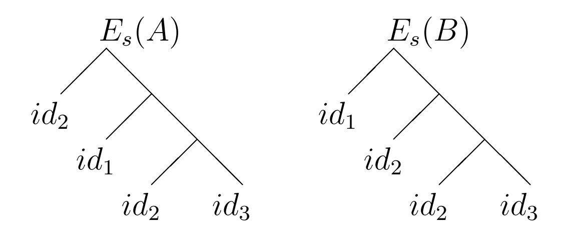
1. $E_s(E) = \{E\}$ für eine Entität E, die keine reinterpretierte Relation ist.

2. Für eine Relation R ist die Multimenge

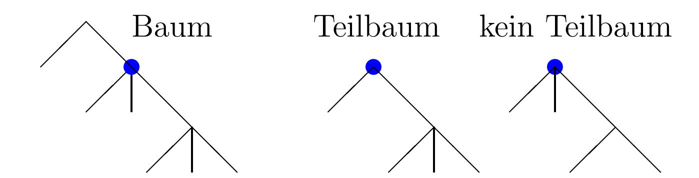
$$E_s(R) = \{E_s(E_i) | R \text{ definiert über } E_i,\ i = 1, \ldots, n\},$$

Wiederholungen von Elementen sind also erlaubt. Wie bereits oben angedeutet, lässt sich diese Definition leicht auf das ERM übertragen und ergibt eine geschachtelte Struktur von Fremdschlüsseln.

Die Definition des Ähnlichkeitsmaßes nutzt aus, dass die $E_s(R)$ einen Baum bilden. Die Bäume zu A und B aus dem Beispiel sind



Das Ähnlichkeitsmaß $A_s(R_1, R_2)$ für zwei Relationen R_1 und R_2 ist dann definiert als der größte Teilbaum von $E_s(R_1)$, der auch Teilbaum von $E_s(R_2)$ ist. Dabei ist ein Teilbaum eines Baumes gegeben durch einen ausgewählten Knoten des Baumes, eine Teilmenge der Nachfolger dieses Knotens, sowie den gesamten Unterbäumen an den ausgewählten Nachfolgern. Die Größe eines Teilbaums meint die Anzahl seiner Blätter. Zur Verdeutlichung (der blaue Knoten ist der ausgewählte):



Die Entscheidung, den zweiten Unterbaum nicht als Teilbaum im Sinne von Strukturähnlichkeit zuzulassen, lässt sich aus der Sicht der Datenmodelle begründen. Es soll nämlich zu einer Relation eine Teilrelation betrachtet werden, die auf weniger Entitäten definiert ist. Sind diese Entitäten reinterpretierte Relationen, dann sollen diese komplett in das Teilmodell aufgenommen werden. Täte man dies nicht, so wäre die Interpretation einer Strukturähnlichkeit sehr erschwert. Außerdem gilt ja, dass der zweite Unterbaum immer noch untersucht werden kann, indem einfach der blaue Knoten eine Stufe tiefer gewählt wird.

Zurück zum Beispiel. Die Teilbäume von $E_s(A)$ außer $E_s(A)$ selbst sind:

$$\{id_1\},\ \{id_2\},\ \{id_3\},\ \{id_2, id_3\},\ \{id_1, \{id_2, id_3\}\},$$

die von $E_s(B)$ hingegen

$$\{id_1\},\ \{id_2\},\ \{id_3\},\ \{id_2, id_3\},\ \{id_2, \{id_2, id_3\}\}.$$

Der größte gemeinsame Teilbaum ist also

$$A_s(A, B) = \{id_2, id_3\}.$$

Daraus lässt sich auch ein numerischer Wert gewinnen, indem zunächst jeder geschachtelten Menge E_s oder A_s eine Multimenge durch Verflachung zugeordnet wird:

$$E_s \mapsto \hat{E}_s := \{E_i | E_i \text{ ist Blatt in } E_s\}.$$

Damit wird definiert

$$a_s(R_1, R_2) := \frac{|\hat{A}_s(R_1, R_2)|}{|\hat{E}_s(R_1) \cup \hat{E}_s(R_2)|}.$$

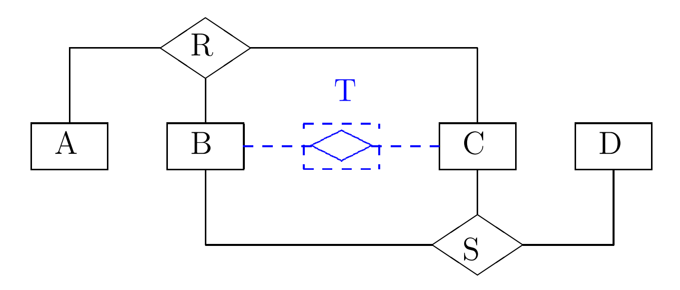
Für das Beispiel wird

$$a_s(A, B) = \frac{|\{id_2, id_3\}|}{|id_1, id_2, id_2, id_3|} = \frac{1}{2}.$$

Die beiden Relationen werden also nicht mehr als strukturidentisch angesehen.

2.2.3 Anwendung: intermediäre Relationen

Das neue Ähnlichkeitsmaß kann Relationen entdecken, die im Datenmodell nicht vorhanden sind, aber als zusätzliche Relationen Sinn machen, weil sie als Aggregation in vorhandenen Relationen vorkommen. Dazu ein Beispiel:



Man erhält

$$\begin{aligned} E_s(R) &= \{A, B, C\} \\ E_s(S) &= \{B, C, D\} \\ A_s(R, S) &= \{B, C\} \end{aligned}$$

Die R und S gemeinsame Teilstruktur entspricht einer neuen intermediärenRelation $T \subseteq B \times C$, die definiert werden kann durch

$$T = \{(b, c) \mid \exists a : (a, b, c) \in R \vee \exists d : (b, c, d) \in S\}$$

Es ist gut vorstellbar, dass in konkreten Datenmodellen diese neue Relation fachlich interpretierbar ist und daher tatsächlich im Datenmodell modelliert werden sollte.

2.2.4 Strukturanalogien zwischen entfernten Teilen eines Datenmodells

Bisher wurden Relationen nur dann als strukturähnlich angesehen, wenn sie direkt oder indirekt zumindest teilweise über denselben Entitäten definiert sind. Das in diesem Abschnitt vorgestellte Ähnlichkeitsmaß kann aber auch Analogien zwischen Relationen aufdecken, die diese Voraussetzung nicht erfüllen, sondern vielleicht sogar aus zwei völlig verschiedenen Datenmodellen stammen. Zwei Relationen sollen dann strukturähnlich heißen, wenn sie gemeinsame Teilbäume besitzen, die die gleiche topologische Struktur haben. Das bedeutet, dass die Etikettierung der Blätter mit Entitätsnamen oder Schlüsseln vernachlässigt wird. Die daraus resultierende Struktur zu einer Relation soll *anonyme Struktur* heißen und mit $E_a(R)$ bezeichnet werden. Man erhält dann daraus das anonyme Ähnlichkeitsmaß $A_a(R_1, R_2)$.

Im Beispiel sieht das dann so aus, dass

$$E_a(A) = E_a(B) = \{\circ, \{\circ, \{\circ, \circ\}\}\}$$

ist – $\circ$ steht dabei für ein Blatt, das sozusagen anonym bleibt. Nun weisen A und B die gleiche topologische Struktur auf.

Diese Vorgehensweise berücksichtigt nicht, dass das Datenmodell in Wirklichkeit kein Baum ist. Im Beispiel etwa sind zwei Blätter identisch. Soll das berücksichtigt werden, d.h. sollen zwei topologische Strukturen nur dann als gleich gelten, wenn identische Blätter in beiden übereinstimmen (bis auf Reihenfolge natürlich), dann ist eine kompliziertere Vorgehensweise nötig.

Ein ausführlicheres Beispiel findet sich in Abschnitt 3.3.2.

Will man zwei verschiedene Datenmodelle auf Analogien untersuchen, so kann man folgenden Trick anwenden: Man führt in jedem Datenmodell eine neue Relation `Global` ein, die formal als Kreuzprodukt aller Relationen des Modells definiert ist. Das Ähnlichkeitsmaß $A_a(Global_1, Global_2)$ liefert dann die größte gemeinsame Teilstruktur zurück. Ein Beispiel dazu findet sich ebenfalls auf in Abschnitt 3.3.2. Dort wird auch diskutiert, welche Auswirkungen es hat, wenn auf identische Baumblätter geachtet wird, diese also in beiden Strukturen übereinstimmen sollen.

3 Anwendung auf Datenmodelle

Aus der bereits genannten Quelle [SC97] soll nun vor allem ein Datenmodell näher untersucht werden, nämlich das Modell zur Vertriebsabwicklung ([SC97], S. 458f, Abb. B.II.25.).

Das Datenmodell liegt als PERM vor, weshalb hauptsächlich der Ähnlichkeitswert für PERM verwendet wird.

Die Komplexität des Modells ist aus der Abbildung 2 ersichtlich, in der die Entitäten/Relationen durch ihre Nummern repräsentiert sind und die gerichteten Pfeile angeben, von welchen Entitäten welche Relation abhängig ist, sie zeigen immer von der Relation auf eine Entität, geben also Fremdschlüsselbeziehungen wieder.

Auch in dieser Darstellung ist klar erkennbar, was Entitäten, was Relationen und was als Entitäten reinterpretierte Relationen sind: Entitäten sind Senken, Relationen Quellen im gerichteten Graphen, alle anderen Knoten sind reinterpretiert.

Es ist hier nicht der Ort, die fachlichen Hintergründe dieses Modells auszubreiten, es sollte der gesunde Menschenverstand ausreichen, um mit den recht klaren Entitätsnamen auch eine einigermaßen klare Auffassung von der sachlichen Bedeutung zu bekommen[2].

Die Zuordnung der Nummern in Abbildung 2 zu den Entitätstypen und Relationen des Modells entnimmt man folgender Tabelle:

Nr.	Entität/Relation	Nr.	Entität/Relation
1	Organisationseinheit	31	Ladeliste
2	Zeit	32	Ladelisteposition
3	Lagerort	33	Versandtour
4	Artikel	34	Kundenrechnung
5	Kundenauftragstyp	35	Kundenrechnungsposition
6	Kunde	36	Versandtransporteinheit
7	Kundenkondition	37	Versandtransportmittel
8	Prüfplan	38	Versandpackungsart
9	Arbeitsplan	39	Transport_Zuo
10	Sachkonto	40	Transportzusammenstellung
11	Debitorenkonto	41	Packung_Zuo
12	Tour	42	Teilstrecke
13	Transporteinheit	43	Kundenanfrage_Angebot_Zuo
14	Transportmittel	44	Kundenauftragstyp_Zuo
15	Packungsart	45	Bedarfsableitung
16	Ort	46	Kundenauftrag_Angebot_Zuo
17	Lagerbestand	47	KundenAbgesauftragstyp_Zuo
18	Kundenanfrage	48	Bedarfsdeckung
19	Kundenanfrageposition	49	Folgeauftrag
20	Lagerdeckung	50	Kundenkonditionen_Zuo
21	Kundenangebot	51	Lager_Liefer_Zuo
22	Kundenangebotsposition	52	Teillieferung
23	Fertigungsauftrag	53	Buchung
24	Bedarfsposition	54	Prüfergebnis
25	Kundenauftrag	55	Packung
26	Kundenauftragsposition	56	Lieferschein_Rechnung_Zuo
27	AbgesKundenauftrag	57	Sachbuchung
28	AbgesKundenauftragsposition	58	Debitorenbuchung
29	Lieferschein	59	Debitoren_Zuo
30	Lieferscheinposition	60	Tour_Zuo

[2] „_Zuo“ steht für „Zuordnung“

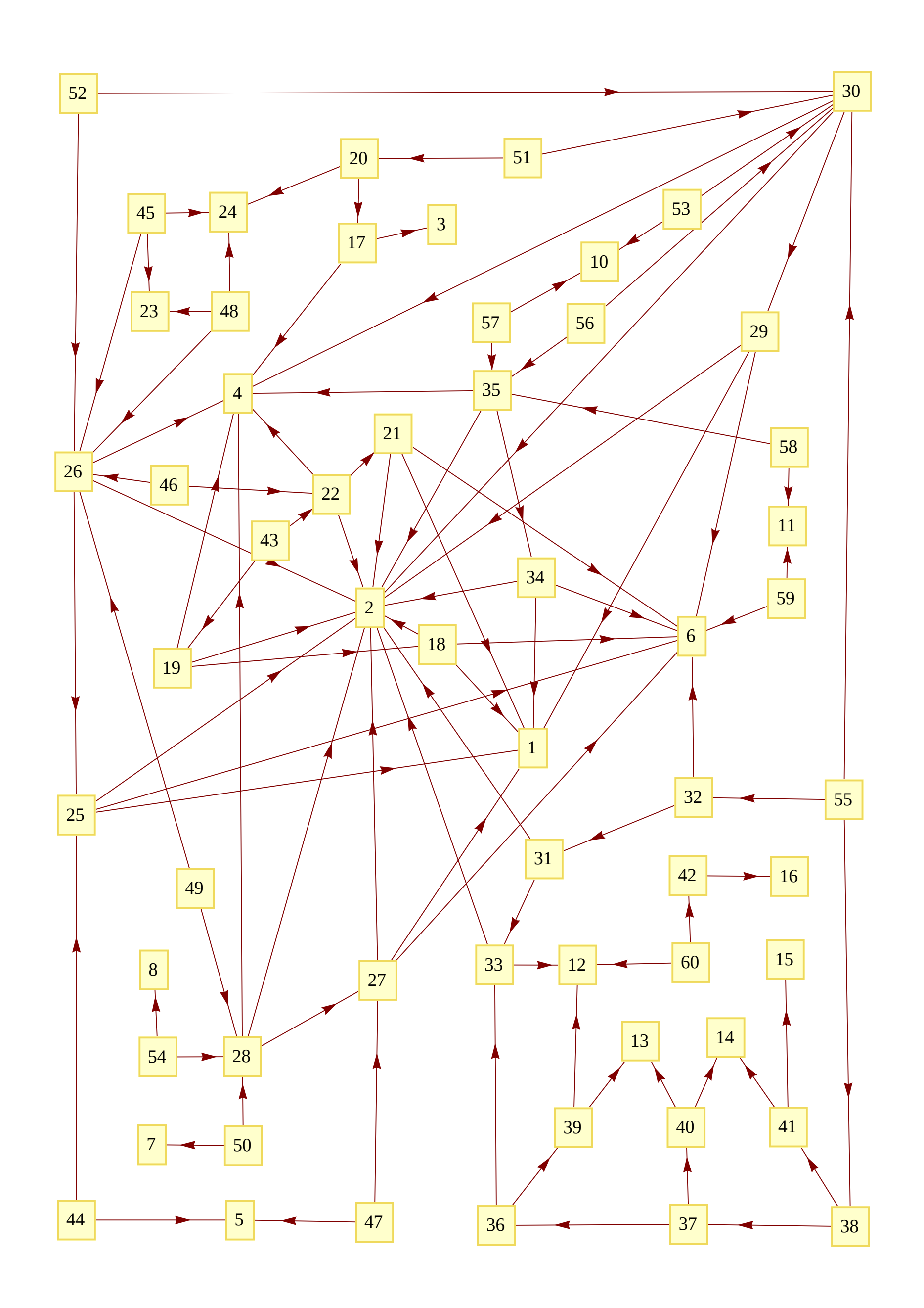


Abbildung 2: *Modell für den Vertrieb, Originalversion*

3.1 Ähnlichkeitswerte

Der Ähnlichkeitswert ist 1 für folgende Relationen:

```
(Kundenanfrage = Kundenangebot
               = Kundenauftrag
               = AbgesKundenauftrag
               = Lieferschein
               = Kundenrechnung)
(Kundenanfrageposition = Kundenangebotsposition
                       = Kundenauftragsposition
                       = AbgesKundenauftragsposition
                       = Lieferscheinposition
                       = Kundenrechnungsposition)
(Kundenauftrag_Angebot_Zuo = Folgeauftrag
                           = Teillieferung
                           = Kundenanfrage_Angebot_Zuo
                           = Lieferschein_Rechnung_Zuo)
(Kundenauftragstyp_Zuo = KundenAbgesauftragstyp_Zuo)
(Bedarfsableitung = Bedarfsdeckung)
(Buchung = Sachbuchung)
```

Diese Gruppen enthalten jeweils strukturgleiche Relationen.

Allerdings ist `Kundenanfrage_Angebot_Zuo` eine 1:n-Relation, während die anderen aus der Gruppe n:m-Relationen sind!

Die nächsthöchsten Ähnlichkeiten sind:

```
d(Kundenauftrag_Angebot_Zuo, Lager_Liefer_Zuo) = 0.75,
d(Folgeauftrag, Lager_Liefer_Zuo) = 0.75,
d(Lager_Liefer_Zuo, Teillieferung) = 0.75.
```

Werden die strukturidentischen Relationen identifiziert, so reduziert sich das PER-Modell ganz erheblich, wie in Abbildung 3 auf der nächsten Seite gezeigt.

Jede Gruppe strukturidentischer Relationen trägt dabei als Label die kleinste Nummer aus der Gruppe. Der Gewinn an Übersichtlichkeit ist frappierend. Ohne Einsatz von Hilfsmitteln hätte diese Reduktion wohl nicht erreicht werden können.

3.2 Beispiel für Unterschiede zwischen PERM und ERM

Im Datenmodell für die Vertriebsabwicklung kommen einige 1:n-Relationen vor, die auf einer als Entität reinterpretierten Relation definiert sind. Hier sollte sich ein deutlicher Unterschied in den Ähnlichkeitskoeffizienten zeigen. Es soll ein Ausschnitt aus dem Datenmodell sowohl im PERM als auch im vereinfachten ERM untersucht werden. Der Ausschnitt enthält alle Entitäten und Relationen, die Abhängigkeiten zu `Ladeliste` und/oder `Versandtransporteinheit` haben. Aufgenommen werden alle Relationen, die direkt oder indirekt von den eben genannten Entitäten abhängen, sowie alle Entitäten und Relationen, auf denen sie direkt oder indirekt definiert sind. Man erhält so 12 Relationen:

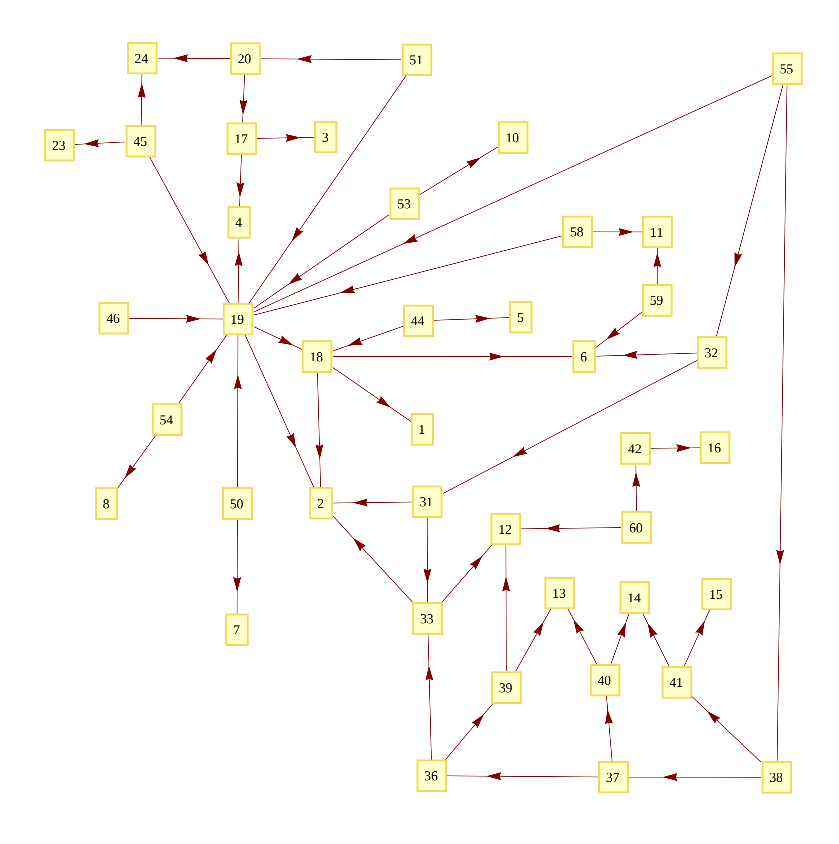


Abbildung 3: *Modell für den Vertrieb, mit Zusammenlegung strukturidentischer Relationen*

1	Lieferscheinposition	7	Transport_Zuo
2	Ladelisteposition	8	Transportzusammenstellung
3	Lieferschein	9	Packungs_Zuo
4	Versandtour	10	Ladeliste
5	Versandtransportmittel	11	Versandtransporteinheit
6	Versandpackungsart	12	Packung

Diese Relationen sind im PERM definiert über den Relationen und Entitäten laut folgender Tabelle:

R	über	R	über
1	3,b,c	7	e,f
2	10,b	8	f,h
3	1,a,b,d	9	g,h
4	a,e	10	4,a
5	8,11	11	4,7
6	5,9	12	1,2,6

a=Zeit, b=Kunde, c=Artikel
d=Organisationseinheit
e=Tour, f=Transporteinheit
g=Packungsart, h=Transportmittel

im PER-Modell sind `Ladeliste` und `Versandtransporteinheit` 1:n-Relationen zwischen `Versandtour` und `Zeit` bzw. zwischen `Versandtour` und `Transport_Zuo`. Im vereinfachten ERM verschwinden diese beiden Relationen, da sie durch Fremdschlüssel in `Versandtour` dargestellt werden. Andere Relationen, die über `Ladeliste` oder `Versandtransporteinheit` definiert sind, werden dann über `Versandtour` definiert.

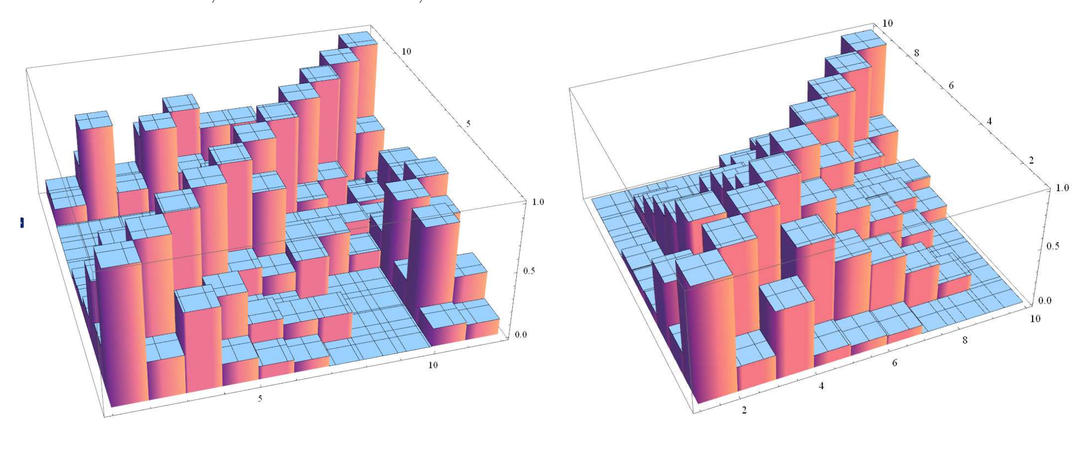
Die Ähnlichkeitsmatrix für den Ausschnitt modelliert mit PERM ist

	1	2	3	4	5	6	7	8	9	10	11	12
1	1.	0.29	0.6	0.17	0.1	0.08	0.	0.	0.	0.14	0.12	0.29
2	0.29	1.	0.4	0.5	0.25	0.2	0.2	0.	0.	0.75	0.33	0.24
3	0.6	0.4	1.	0.25	0.12	0.1	0.	0.	0.	0.2	0.17	0.18
4	0.17	0.5	0.25	1.	0.33	0.25	0.33	0.	0.	0.67	0.5	0.12
5	0.1	0.25	0.12	0.33	1.	0.75	0.33	0.33	0.14	0.29	0.67	0.35
6	0.08	0.2	0.1	0.25	0.75	1.	0.25	0.25	0.25	0.22	0.5	0.47
7	0.	0.2	0.	0.33	0.33	0.25	1.	0.33	0.	0.25	0.5	0.12
8	0.	0.	0.	0.	0.33	0.25	0.33	1.	0.33	0.	0.2	0.12
9	0.	0.	0.	0.	0.14	0.25	0.	0.33	1.	0.	0.	0.12
10	0.14	0.75	0.2	0.67	0.29	0.22	0.25	0.	0.	1.	0.4	0.18
11	0.12	0.33	0.17	0.5	0.67	0.5	0.5	0.2	0.	0.4	1.	0.24
12	0.29	0.24	0.18	0.12	0.35	0.47	0.12	0.12	0.12	0.18	0.24	1.

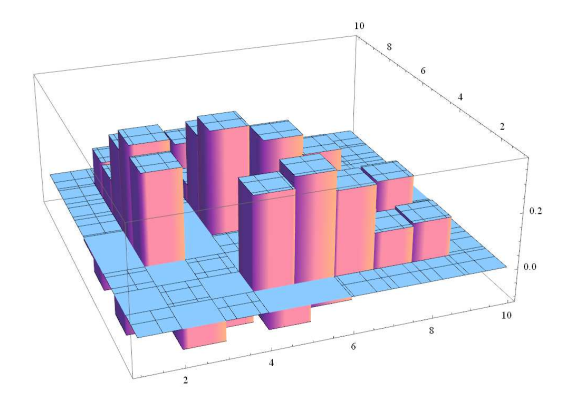
Die Ähnlichkeitsmatrix für den Ausschnitt modelliert mit dem vereinfachten ERM ergibt deutlich veränderte Werte. In der folgenden Tabelle sind die Werte für die nicht mehr vorhandenen Relationen durch „–“ ersetzt worden, damit die beiden Tabellen leichter vergleichbar sind. Man erkennt, dass fast alle Ähnlichkeitswerte verändert werden, meist verringern sie sich, oft aber werden sie auch teils erheblich größer (21 mal verringert, 12 mal erhöht). Der mittlere Ähnlichkeitswert erhöht sich von 0.2828 auf 0.3168[3]:

	1	2	3	4	5	6	7	8	9	10	11	12
1	1.	0.22	0.6	0.11	0.09	0.08	0.	0.	0.	–	–	0.25
2	0.22	1.	0.29	0.83	0.62	0.5	0.33	0.14	0.	–	–	0.3
3	0.6	0.29	1.	0.14	0.11	0.09	0.	0.	0.	–	–	0.15
4	0.11	0.83	0.14	1.	0.71	0.56	0.4	0.17	0.	–	–	0.25
5	0.09	0.62	0.11	0.71	1.	0.78	0.29	0.29	0.12	–	–	0.35
6	0.08	0.5	0.09	0.56	0.78	1.	0.22	0.22	0.22	–	–	0.45
7	0.	0.33	0.	0.4	0.29	0.22	1.	0.33	0.	–	–	0.1
8	0.	0.14	0.	0.17	0.29	0.22	0.33	1.	0.33	–	–	0.1
9	0.	0.	0.	0.	0.12	0.22	0.	0.33	1.	–	–	0.1
10	–	–	–	–	–	–	–	–	–	–	–	–
11	–	–	–	–	–	–	–	–	–	–	–	–
12	0.25	0.3	0.15	0.25	0.35	0.45	0.1	0.1	0.1	–	–	1.

Fürs Auge schön anzuschauen sind die grafischen Darstellungen dieser Matrizen. Zuerst Ähnlichkeiten nach PERM, dann nach ERM, und abschließend die Differenzen:



[3]wobei nur Entitäten berücksichtigt sind, die in beiden Modellen vorkommen



3.3 Anwendung des strukturierten Ähnlichkeitsmaßes

3.3.1 Intermediäre Relationen

Eine ausführliche Suche im Vertriebsmodell nach intermediären Relationen erbrachte drei Kandidaten:

(Organisationseinheit, Zeit) als gemeinsame Teilstruktur von z.B. `Bedarfsableitung` und `Beschaffungsbeleg` oder von `Bedarfsableitung` und `Beschaffungsbelegposition`, sowie an vielen anderen Stellen;

(Teil, Zeit) als gemeinsame Teilstruktur von z.B. `Maschinenbelegung` mit `Auftragsposition`, `Lagerdeckung` oder `Lager_Liefer_Zuo`;

(Zeit, (Organisationseinheit, Organisationseinheit, Zeit)) als gemeinsame Teilstruktur von z.B. `Werkzeugbelegung` und `Mitarbeiterbelegung`, `MitarbeiterMaschinenbelegung` oder `Mitarbeiter_Unterbrechung_Zuo`.

Insbesondere die letzte mit ihrer komplexen Struktur scheint einer näheren Betrachtung wert zu sein. Die Einführung einer neuen Relation für diese Struktur könnte sich lohnen und fachlich interpretierbar sein.

3.3.2 Strukturanalogien

Das Modell zur Vertriebsabwicklung ist reich an strukturidentischen Relationen. Es ist zu erwarten, dass eine Suche nach Strukturanalogien mit dem Ähnlichkeitsmaß ohne Berücksichtigung der Entitätsnamen (reine topologische Ähnlichkeit, anonyme Ähnlichkeit) lohnend ist.

Neben den bereits bestehenden Strukturidentitäten kommen nun hinzu:

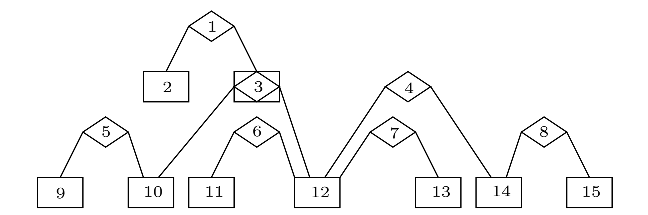
- **Versandtour, Lagerbestand, Debitoren_Zuo, Transport_Zuo, Transportzusammenstellung, Packung_Zuo, Teilstrecke** mit der Struktur $(\circ, \circ)$;
- **Ladeliste, Tour_Zuo, Lagerdeckung** mit der Struktur $(\circ, (\circ, \circ))$;
- **Kundenkonditionen_Zuo, Prüfergebnis, Buchung, Sachbuchung, Debitorenbuchung** mit der Struktur $(\circ, (\circ, \circ, (\circ, \circ, \circ)))$. Vorher waren in dieser Gruppe nur *Buchung* und *Sachbuchung* enthalten.

Die Erwartung wird allerdings enttäuscht, denn erstens ergeben sich nur wenige neue Analogien, und zweitens ist keineswegs klar, welchen Nutzen man aus der – sehr einfachen – gemeinsamen Struktur ziehen soll.

3.3.3 Gleiche Teilmodelle in verschiedenen Datenmodellen

Der erwähnte Trick, in zwei Datenmodellen das größte gemeinsame topologisch gleiche Teilmodell zu finden, ist aufwändig. Deshalb soll hier nur kurz über das Ergebnis berichtet werden, wenn man ein Modell zur Bedarfsplanung etc. mit einem Modell zur Beschaffungslogistik vergleicht ([SC97], S. 176, Abb. B.I.66, S. 254f, Abb. B.I.133, S. 418f, Abb. B.II.07).

Eine von Hand durchgeführte Suche nach dem größtmöglichen isomorphen Untergraphen in beiden Datenmodellen, der natürlich zu jeder Relation auch alle die Entitäten enthalten muss, auf denen sie definiert ist, führte zu folgendem Teilmodell:



Für die Benennung der Entitäten und Relationen gibt es in beiden Datenmodellen mehrere Möglichkeiten. Eine Möglichkeit, den Nummern Namen zuzuordnen, ist in der folgenden Tabelle gezeigt:

lfd. Nr.	Beschaffungslogistik	Bedarfsplanung etc.
9	Lieferantenmerkmal	TechnischesVerfahren
5	Lieferantenmerkmal_Zuo	Verfahren_Zuo
2	Periodenraster	Arbeitsplatzgruppe
1	Zeitreihe_2	Arbeitsgang_Zuo
10	Lieferant	Standardarbeitsgang
3	Lieferantenkondition	Arbeitsgang
11	Materialmerkmal	Reihenfolge
6	Materialmerkmal_Zuo	Arbeitsplanreihenfolge
12	Fremdgut	Arbeitsplan
7	Fremdgut_Text_Zuo	Arbeitsplan_Zuo
13	Lieferantentext	Teil
4	Fremdgut_Zuo	Arbeitsplan_Werk_Zuo
14	Einkaufsgruppe	Werk
8	Einkäufergruppierung	Werksgruppierung
15	Einkäufer	Werksbereich

Man erkennt deutliche semantische Übereinstimmungen, z.B. bei den Nummern 10, 3, 11, 6, 12, 7, 13 und 4; dabei ist der Abschnitt 12, 7, 13 besonders eindringlich. Ebenso bei 14, 8, 15. Hier zeigen sich in gewisser Weise *Patterns* der Datenmodellierung.

4 Implementierung

Die beschriebenen Verfahren zur Berechnung von Ähnlichkeitswerten und zur Suche nach strukturidentischen Relationen wurden mit Mathematica® implementiert. Der Quellcode liegt in Form eines Mathematica-*Notebooks* vor. Die Verwendung erfolgt dadurch, dass das Notebook in einer Mathematica-Sitzung ausgeführt wird.

Funktionen zur Analyse des Datenmodells

Das Datenmodell wird übergeben in einer „Arrows" genannten Datenstruktur. Aufbau: {{r1,e1},{r2,e2},{r3,e3},...}. Jeder Eintrag {r1,e1} entspricht einer Relation r1, die über der Entität e1 definiert ist. Ist eine Relation über mehreren Entitäten definiert, so gibt es entsprechend viele Einträge. Parallele Beziehungen führen zu identischen Einträgen.

Beschreibung der Funktionen

```
nestedEntitiesOfRelations
Rückgabe: Liste der Gestalt {{r1,ents1}{r2,ents2},...}, bestehend aus
Paaren mit einer Relation (erstes Element) und der geschachtelten
Entitätsstruktur, auf der die Relation definiert ist (zweites
Element).
Argument ist die Liste der Arrows.

entitiesOfRelations
Rückgabe: Liste der Gestalt {{r1,ents1}{r2,ents2},...}, bestehend aus
Paaren mit einer Relation (erstes Element) und der Liste der
Entitäten, auf der die Relation definiert ist (zweites Element).
Argument ist die Liste der Arrows.

structuredIdentities
Rückgabe: Liste, bestehend aus Mengen strukturidentischer Relationen
(erstes Element) und der geschachtelten Entitätsstruktur, auf der die
Relation(-enmenge) definiert ist (zweites Element).
Strukturidentität beachtet Schachtelung.
Argument ist die Liste der Arrows.

unstructuredIdentities
Rückgabe: Liste, bestehend aus Mengen strukturidentischer Relationen
(erstes Element) und der Liste der Entitäten, auf der die
Relation(-enmenge) definiert ist (zweites Element).
Strukturidentität beachtet Schachtelung nicht.
Argument ist die Liste der Arrows.

tableSimilarityValues
Rückgabe: Tabelle der unstrukturierten Ähnlichkeitswerte, sortiert
nach Relationsnummern.  Relationsnummern sind nicht Bestandteil der
Tabelle.
Argument ist die Liste der Arrows.

tableStructuredValues
Rückgabe: Tabelle der strukturierten Ähnlichkeitswerte, sortiert nach
```

```
Relationsnummern.  Relationsnummern sind nicht Bestandteil der
Tabelle.
Argument ist die Liste der Arrows.

similarityValue, structuredValue, commonStructure:
Funktionen, die Ähnlichkeitswerte etc. einzeln abfragen.
Argumente sind: Relationsnummer, Relationsnummer, Arrows.
(Nummern müssen nicht konsekutiv sein, sondern so, wie sie in Arrows
vorkommen.)

intermedRelations
Rückgabe: Liste der intemediären Relationen.
Argument ist die Liste der Arrows.
```

Anwendungsbeispiel

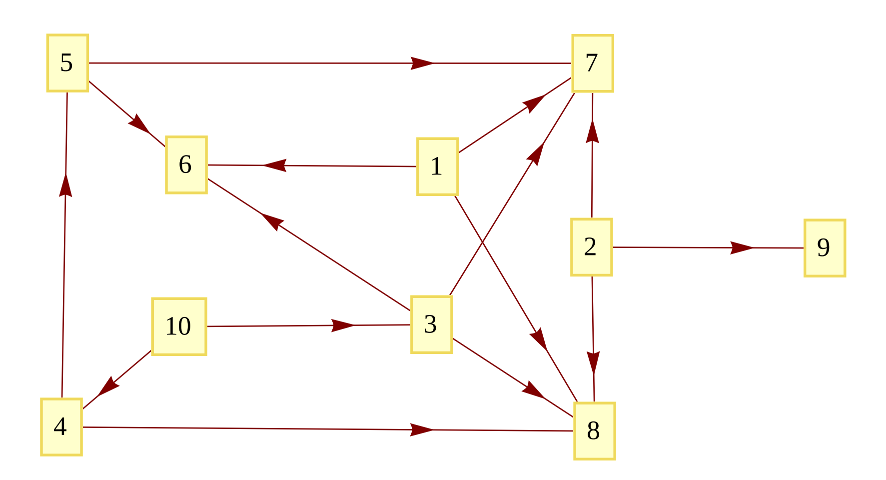
Zunächst die Definition eines Datenmodelles über eine Liste von „Arrows“, das Datenmodell wird anschließend grafisch dargestellt.

```
arrs1 = {{1, 6}, {1, 7}, {1, 8}, {2, 7}, {2, 8},
   {2, 9}, {10, 3}, {10, 4}, {3, 6}, {3, 7}, {3, 8},
   {4, 8}, {4, 5}, {5, 6}, {5, 7}};

graph1 = Rule @@ # & /@ arrs1

{1 → 6, 1 → 7, 1 → 8, 2 → 7, 2 → 8, 2 → 9, 10 → 3,
 10 → 4, 3 → 6, 3 → 7, 3 → 8, 4 → 8, 4 → 5, 5 → 6, 5 → 7}

GraphPlot[graph1, VertexLabeling → True,
 DirectedEdges → True]
```



Es folgen verschiedene Kommandos zur Untersuchung der Struktur des Datenmodells, zur Aufdeckung von Strukturanalogien und zur Berechnung von Ähnlichkeitswerten:

```
rels = (arrs1 // Transpose)[[1]] // Union // Sort
```

{1, 2, 3, 4, 5, 10}

```
commonStructure[1, 5, arrs1]
```

{6, 7}

```
intermedRelations[arrs1]
```

(7 8)

```
entitiesOfRelations[arrs1]
```

1	{6, 7, 8}
2	{7, 8, 9}
3	{6, 7, 8}
4	{6, 7, 8}
5	{6, 7}
10	{6, 6, 7, 7, 8, 8}

```
nestedEntitiesOfRelations[arrs1]
```

1	{6, 7, 8}
2	{7, 8, 9}
3	{6, 7, 8}
4	{8, {6, 7}}
5	{6, 7}
10	{{8, {6, 7}}, {6, 7, 8}}

```
structuredIdentities[arrs1]
```

({1, 3} {6, 7, 8})

```
unstructuredIdentities[arrs1]
```

({1, 3, 4} {6, 7, 8})

```
tableSimilarityValues[arrs1]
```

$$\begin{pmatrix} 1 & \frac{1}{2} & 1 & 1 & \frac{2}{3} & \frac{1}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} & \frac{2}{7} \\ 1 & \frac{1}{2} & 1 & 1 & \frac{2}{3} & \frac{1}{2} \\ 1 & \frac{1}{2} & 1 & 1 & \frac{2}{3} & \frac{1}{2} \\ \frac{2}{3} & \frac{1}{4} & \frac{2}{3} & \frac{2}{3} & 1 & \frac{1}{3} \\ \frac{1}{2} & \frac{2}{7} & \frac{1}{2} & \frac{1}{2} & \frac{1}{3} & 1 \end{pmatrix}$$

```
tableStructuredValues[arrs1]
```

$$\begin{pmatrix} 1 & \frac{1}{2} & 1 & \frac{2}{3} & \frac{2}{3} & \frac{1}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{2}{7} \\ 1 & \frac{1}{2} & 1 & \frac{2}{3} & \frac{2}{3} & \frac{1}{2} \\ \frac{2}{3} & \frac{1}{4} & \frac{2}{3} & 1 & \frac{2}{3} & \frac{1}{2} \\ \frac{2}{3} & \frac{1}{4} & \frac{2}{3} & \frac{2}{3} & 1 & \frac{1}{3} \\ \frac{1}{2} & \frac{2}{7} & \frac{1}{2} & \frac{1}{2} & \frac{1}{3} & 1 \end{pmatrix}$$

Literatur

[BA86] Batini, Carlo; Lenzerini, Maurizio; Navathe, Shamkant B.: A Comparitive Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys 18 (1986) 4, 323-364

[CH76] Chen, Peter Pin-Shan: The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems 1 (1976) 1, S.9-36

[FE05] Fettke, Peter; Loos, Peter: Zur Idenfikation von Strukturanalogien in Datenmodellen. Wirtschaftsinformatik 47 (2005) 2, S. 89-100

[JA02] Helmut Jarosch: Datenbankentwurf - Eine beispielorientierte Einführung für Studenten und Praktiker. Vieweg 2002

[LO97] P. Loos: Capture More Data Semantic Through The Expanded Entity-Relationship Model, Arbeitsbericht des Instituts für Wirtschaftsinformatik, Nr. 53, Münster, März 1997

[MA04] Stephen Wolfram: The Mathematica Book. Wolfram Media, 2004.

[MA09] Wolfram Research: Mathematica Version 7.0.1, März 2009.

[SC97] Scheer, August-Wilhelm: Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftsprozesse. Berlin et al. 1997

Praxisorientierte und innovative Lehre in der Wirtschaftsinformatik

Norbert Tschritter

Am Beispiel von RFID-Projekten wird beschrieben, wie es gelingt, Studierende mit modernen Lehrmethoden durch modulare Themenaufbereitung praxisnah mit interdisziplinärer Zusammenarbeit zu konfrontieren und für neue Technologien zu begeistern.

1 Einleitung

Mit modernen Lehrmethoden wird das Innovationspotential von Radiofrequenzidentifikation (RFID) in verschiedenen Projekten und Seminaren durch eine Koppelung von theoretischer Fundierung und praktischen Arbeiten den Studierenden zugänglich gemacht. Die Studierenden erfahren praxisnah die technischen und kommunikativen Schwierigkeiten.

2 Kurzbeschreibung der RFID

RFID dient der elektronischen Identifikation von Objekten und Lebewesen durch kontaktlose Datenübertragung vom Datenträger (Transponder)[1] zum Lesegerät. Dabei wird meist auch die vom Datenträger benötigte Energie kontaktlos durch das Lesegerät mittels elektromagnetischem Feld erzeugt.[2] Die Reichweite der Systeme ist abhängig von der Betriebsfrequenz und reicht üblicherweise von wenigen Millimetern bis zu 15 Metern.[3] Den besten Kompromiss aus Reichweite und Übertragungsgeschwindigkeit

---

[1] Siehe: Manish Bhuptani, Sharam Moradpour, S. 24, Transponder (abgeleitet von Transmitter und Responder), auch Tag genannt.

[2] vgl. Martin Vincenz, S. 70 f.

[3] vgl. BSI, S. 29 ff.

bieten Frequenzen im Kurzwellenbereich. Reichweiten von 10 – 15 Metern werden meist mit Frequenzen im Mikrowellenbereich und mit aktiven (batteriegespeisten) Tags realisiert.[4] RFID-Systeme unterscheiden sich hauptsächlich in der Art der Energieversorgung, der Speichertechnologie und der Betriebsfrequenz. Transponder können dabei die unterschiedlichsten Bauformen aufweisen, z. B. Münzen, Glasgehäuse, Chipkarten, Uhren, Smart-Labels.[5] Ein RFID-System besteht somit aus dem Transponder, der Schreib-Lese-Einheit und der angeschlossenen Datenverarbeitung.

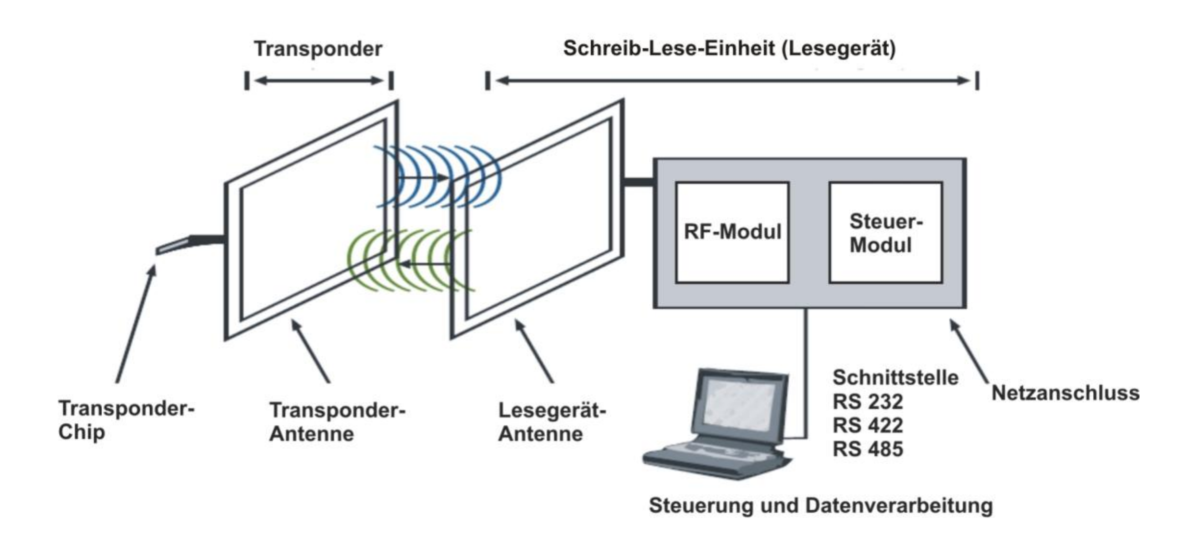


Abb. 1: Aufbau und grundsätzliche Funktion eines RFID-Systems[6]

Durch die Vernetzung verschiedener RFID-Systeme und angeschlossener Datenbanken werden die markanten Vorteile aktuell hauptsächlich in einem effizienteren Liefernetzwerk und/oder im effizienteren Produktionsmittelmanagement gesehen. Eine Optimierung der Steuerung von Lieferketten wird durch Verfahren zur Reduzierung der Lagermengen oder durch eine lückenlose Warenverfolgung ermöglicht.[7]

Durch zunehmende unternehmensweite Zusammenarbeit und der damit verbundenen IT-Integration wird zwar eine höhere Effizienz erzielt, allerdings ist der Preis dafür ein oft ungewollt höheres Risiko des Datenmissbrauchs und aufgrund fehlender

[4] Vgl. Robert und Gabriele Schoblick, S. 127 ff.

[5] Vgl. Klaus Finkenzeller, S. 1, 13 ff.

[6] Vgl. X-ident technology, http://www.x-ident.com/deutsch/technologie.htm, Stand: 10.02.10, 16:45
auch verwendet von: [2] Martin Vincenz, S. 71 und vom [3] BSI, S. 21, Abb. 4-1

[7] vgl. Martin Strassner, Innovationspotential von RFID für das Supply Chain Management, http://www.alexandria.unisg.ch/Projekte/einfache-Suche/18175, Stand: 05.01.2010, 10:26

Kontrollmechanismen ein damit einhergehender Vertrauensverlust in die neuen Technologien[8].

3 Lehren, das Lernen zu lernen

Vor dem Hintergrund der sich schnell ändernden Anforderungen im Berufsleben – besonders in der Informationstechnologie – ist ein Lernen nach starrem Muster nicht zielführend. Es erfordert die Fähigkeit, flexibel auf neue Situationen und Anforderungen reagieren zu können - es erfordert: „Das Lernen zu lernen“. Lernen wird durch eine aktive Beteiligung der Studierenden an den Lehrzielen und an der Art und Weise der Zielerreichung zu einem aktiven Prozess. Die Studierenden kennen die (ihre) Lernziele, erkennen Lernfortschritte und sind in der Lage das Erlernte entsprechend den Projektanforderungen in einem sinnvollen Kontext anzuwenden.[9] Sie studieren selbstständig.[10]

4 Innovative Lehre am Beispiel RFID

Besonderes Augenmerk verdiente die interdisziplinäre Zusammenarbeit in BWL-IT-Projekten, in denen Studierende unterschiedlicher Studiengänge gemeinsam sowohl technische als auch organisatorische Probleme entsprechend ihrer jeweiligen Fachrichtungen zu bewältigen hatten. In diesen Projekten war das Engagement der Studierenden sehr hoch. Wegen der modularen Themenaufbereitung und der damit entstandenen inhaltlichen und zeitlichen Abhängigkeiten der Arbeitsgruppen voneinander war aber auch das Konfliktpotential nicht zu unterschätzen. Das Zerlegen eines Projektes in kleinere Projektteile bewirkte, dass sich die Studierenden mit dem Teilgebiet besonders intensiv auseinandersetzten, welches ihren jeweiligen Neigungen am meisten entsprach. Die Aufgabenteilung machte zum Projektabschluss eine Zusammenfügung der Teilprojekte zu einem Ganzen notwendig und erzwang eine entsprechende Koordination und Schnittstellenanpassung. Das erhöhte die Komplexität der Projekte, sorgte aber auch für einen Problem- und Erfahrungsaustausch der Gruppen untereinander, um die Projekte erfolgreich zu beenden.

Neben den Koordinations- und Schnittstellenproblemen wurden die Studierenden der Wirtschaftsinformatik mit ganz speziellen Herausforderungen und häufig anfangs nicht absehbaren Problemen der RFID-Technologie konfrontiert. Die Unterschiede von Theorie

---

[8] vgl. Philipp Bensel, Markus Richter, Stefan Vogeler, Diffusion von Vertrauen in unternehmens-übergreifenden RFID-Anwendungen http://opus.kobv.de/tuberlin/volltexte/2008/1937/pdf/bensel_philipp_5_2008.pdf, Stand: 05.01.2010, 10:55

[9] vgl. Herman Blom, S. 12 ff.

[10] siehe auch: Kretschmar, Plietz, Kap. 2.1, S. 12

(Beschreibung einer Technik) und Praxis (Umgang mit der Technik) haben die Studierenden dabei hautnah und manchmal auch leidvoll erlebt.

Die Ursachen dafür lagen in der hohen technischen Komplexität und in der Neuheit der Anwendungen. Mangels Alternativen musste oft mit Beta-Software programmiert werden, mit der nicht unerwarteten Folge, sich auch noch mit Software-Bugs beschäftigen zu müssen. Somit war es nicht erstaunlich, dass z. B. Dokumentationen und Handbücher in diesen Fällen keine Hilfestellung mehr bieten konnten. Mit Einfallsreichtum und Kreativität haben die Studierenden die technischen Probleme gelöst oder sie geschickt umgangen.

Es ist jedoch zu bemerken, dass sich die Studierenden in allen Projekten sehr intensiv mit ihrer jeweiligen Rolle innerhalb der Projekte identifiziert haben und dass ein Interesse für die Projekt- oder Seminarthemen nicht selten über die Veranstaltungszeit hinaus geweckt wurde.

5 Projektbeschreibungen

Im Folgenden werden exemplarisch Projekte unterschiedlicher Seminare beschrieben, die neue Möglichkeiten zur automatischen Identifikation, Dokumentation und zur mobilen Datenspeicherung erproben. Zusätzliche Informationen und Beschreibungen sind zu allen Projekten unter dem jeweiligen Projektnamen auf der beigefügten CD verfügbar.

5.1 Projekt TranspoFil – WS 2005/2006

Das Softwareprojekt **TranspoFil** war ein Pilotsystem das die RFID-Technologie zur Identifizierung und Verwaltung von Dokumenten nutzte. Dabei konnte **TranspoFil** sowohl mit der NTFS-Stream-Technologie als auch mit der Hostscript-Technologie eingesetzt werden. Es ermöglichte, das elektronische Äquivalent eines Dokumentes mit einem RFID-Transponder als Stellvertreter für das physikalische Dokument zu verknüpfen. Außerdem wurden die Bearbeitungsvorgänge eines Dokumentes auf dem RFID-Transponder und der Datei gespeichert.

5.2 Projekt TranspoFil2 – WS 2006/2007

Dieses Projekt beschrieb den Einsatz von RFID-Technologien in der Verwaltung von Wasch- und Trocknungsvorgängen und zeigte Einsatzmöglichkeiten für Transponder, die auf der Read-Write-Speichertechnologie basierten. Es wurden spezielle Informationen über Kleidungstücke sowie Wasch- und Trocknungsgänge auf dem Transponder gespeichert und zur Optimierung von Wasch- und Trocknungsprozessen verwendet. Im Rahmen des Softwareprojektes **TranspoFil2** wurde ein Pilotsystem entwickelt und implementiert, welches die Ergebnisse des beschriebenen Einsatzes präsentiert.

5.3 Projekt IWAS 1.0 - SoSe 2007

Das Projekt **IWAS 1.0** stand für die Bezeichnung „intelligentes Waschen“ Version 1.0. Das System wurde im Rahmen der Veranstaltung „Anwendungsentwicklung 2“ entwickelt**. IWAS 1.0** baute auf „Transprofil 2“ auf. Es wurden jedoch nur vereinzelt Funktionen aus diesem Projekt modifiziert und genutzt. Ziel war die Optimierung der Waschvorgänge einer Wäscherei. Mehrere Transponder konnten gleichzeitig gelesen, beschrieben und ausgewertet werden. Dabei wurde davon ausgegangen, dass mehrere Kleidungsstücke sich in dem Bereich einer RFID- Antenne befinden können. Jeder Transponder der Kleidungsstücke wurde gelesen und enthielt die notwendigen Informationen zur Ermittlung des optimalen Waschprogramms. Bei der Eingabe der Attribute standen die genormten Wäschesymbole zur Verfügung. Das Setzen der Werte war dem Benutzer oder der Benutzerin auch ohne Detailwissen möglich. Dazu musste lediglich auf die abgebildeten Symbole geklickt werden, welche dem Label des zu registrierenden Kleidungsstückes entsprach.

5.4 Projekt iVis – SoSe 2007

Im Projekt iVis (intelligentes validierbares integriertes Serviceheft) ging es um die RFID-gestützte Fortschreibung von Dokumenten. Das Vertrauen in Schriftstücke wurde gesteigert, weil Manipulationen leichter entdeckt werden können. Exemplarisch wurden dazu Fahrzeug-Servicehefte herangezogen. Zusätzlich zur bislang üblichen manuellen Fortschreibung des Heftes wurden die Serviceeinträge auf einem Transponder gespeichert, der in das Serviceheft geklebt bzw. eingearbeitet wurde. Die Überprüfung der Validität der Einträge wurde durch das zum Fortschreiben der Transponder synchrone Pflegen einer zentralen Datenbank ermöglicht.

6 Literaturverzeichnis

[1] Manish Bhuptani, Sharam Moradpour, RFID Field Guide, Deploying Radio Frequency Identification Systems, Sun Microsystems Press, 2005, ISBN 0-13-185355-4

[2] Martin Vincenz, Möglichkeiten und Grenzen heutiger Transpondertechnologien in der Logistik, VDI-Berichte Nr. 1744, 2003

[3] Bundesamt für Sicherheit in der Informationstechnik, Risiken und Chancen des Einsatzes von RFID-Systemen, Bonn, 2005, http://www.rfidconsultation.eu/docs/ficheiros/RIKCHA.pdf, Stand: 02.02.2010, 16:40

[4] Robert und Gabriele Schoblick, RFID, Radio Frequency Identification, Franzis Verlag, Poing, 2005, ISBN 3-77235920-5

[5] Klaus Finkenzeller, RFID Handbuch, Grundlagen und praktische Anwendungen von Transpondern, kontaktlosen Chipkarten und NFC, 5. Auflage, Carl Hanser Verlag München, 2008, ISBN 978-3-446-41200-2

[6] X-ident technology GmbH, Kreuzauer Strasse 33, 52355 Düren, Germany, http://www.x-ident.com

[7] Martin Strassner, Innovationspotential von RFID für das Supply Chain Management, http://www.alexandria.unisg.ch/Projekte/einfache-Suche/18175, Stand: 05.01.2010, 10:26

[8] Philipp Bensel, Markus Richter, Stefan Vogeler, Diffusion von Vertrauen in unternehmens-übergreifenden RFID-Anwendungen, Frank Straube (Hrsg.), Digitale Schriftenreihe Logistik Technische Universität Berlin, Universitätsverlag der Technischen Universität Berlin, ISSN 1865-5726, Nr. 5, Mai 2008, http://opus.kobv.de/tuberlin/volltexte/2008/1937/pdf/bensel_philipp_5_2008.pdf, Stand: 05.01.2010, 10:55

[9] Herman Blohm, Der Dozent als Coach, Neuwied; Kriftel, Luchterhand Verlag, 2000, ISBN3-472-04425-X

[10] Werner Kretschmar, Ernst Plietz, Reihe: Gestaltung motivierender Lehre in Hochschulen: Praxisanregungen, Die Vorlesung – eine Anleitung zu ihrer Gestaltung, UVW Der Fachverlag für Hochschulthemen, Universitätsverlag Webler, 2. Auflage 2005, ISBN 3-937026-37-1

Von der Massendatenerfassung zur Klimadatenbank

Projektarbeiten im Fach Anwendungsentwicklung 2

Christian Wagner

Welche Probleme können auftreten, wenn man es in der Informationstechnologie mit großen Datenmengen (sog. Massendaten) zu tun bekommt? Gibt es Unterschiede bei der Verarbeitung, Persistierung und Verwaltung? Mit diesen und weiteren Fragen beschäftigten sich Studierende der Wirtschaftsinformatik in der Veranstaltung "Anwendungsentwicklung 2". In kleinen Projektgruppen von drei bis vier Studierenden wurde versucht, anhand realer Massendaten am Beispiel des Aufbaus einer Klimadatenbank anwendungsorientierte Lösungen zu finden.

1 Einführung

Die Wirtschaftsinformatik beschäftigt sich mit der Gestaltung und dem Betrieb von Systemen der computergestützten Informationsverarbeitung für betriebswirtschaftliche Aufgaben. [1] Die Studierenden der Wirtschaftsinformatik an der Leuphana Universität Lüneburg absolvieren die Vorlesungen „Anwendungsentwicklung" und „Anwendungsentwicklung 2" als Pflichtfach im Hauptstudium. Während in der ersten Veranstaltung die Kenntnisse über Prinzipien, Methoden und Verfahren für den Entwurf, die Konstruktion und den Test von betrieblichen Informationssystemen vertieft werden [2], wird in „Anwendungsentwicklung 2" anhand einer konkreten Projektarbeit innerhalb einer Projektgruppe das gewonnene Wissen angewendet. Den Studierenden steht dabei die Wahl der Programmiersprache und der verwendeten Software nahezu frei.

1.1 Warum benötigen wir Massendaten? => Datenbanken

Bei betriebswirtschaftlichen Anwendungen haben wir es in der Regel mit der Verwaltung von großen Datenmengen (Massendaten) zu tun. Sei es, dass man eine Vereins- oder Mitgliederverwaltung, Kontentmanagement-, Buch- oder Lagerhaltungssoftware programmiert, die Anwendungsszenarien sind mannigfaltig. So finden wir bei fast jeder betriebswirtschaftlich geprägten Programmieraufgabe in irgendeiner Form eine Anbindung an ein Datenbanksystem (DBS).

Ein **Datenbanksystem** (**DBS**) ist ein System zur elektronischen Datenverwaltung. Die wesentliche Aufgabe eines DBS ist es, große Datenmengen effizient, widerspruchsfrei und dauerhaft zu speichern und benötigte Teilmengen in unterschiedlichen,

bedarfsgerechten Darstellungsformen für Benutzer und Anwendungsprogramme bereitzu-stellen.[3]

Ein Beispiel: Eine Studierenden-Gruppe programmiert für eine Arztpraxisverwaltung einen ‘intelligenten Impfausweis’ (unter Verwendung von Chipkarte oder RFID-Transponder). Um das Programm auf Funktionalität und Fehler testen zu können, müssen Patientendaten her.

1.2 Woher bekommen wir Massendaten ?

In obigem Beispiel hat die Studierendengruppe mehrere Möglichkeiten:

- Kontaktaufnahme und Kooperation mit einer Arztpraxis zwecks Patienten-Datenbeschaffung – die Erfolgsaussichten sind aber eher gering, da es sich um sensible Personendaten handelt. Die Datenweitergabe wird durch Datenschutzbestimmungen verhindert.

- Füllen der selbst erstellten Praxisdatenbank mit Dummy-Datensätzen (d.h. erfundenen Personendaten) von einem Mitglied des Projektteams – die Erfolgsaussichten sind hierbei ganz gut, da es sich um eine reine Fleißarbeit handelt. Das Problem ist: Die investierte Zeit fehlt bei der sonstigen Projektarbeit, Datensätze in ein DBS eingeben kann und darf bei einer Prüfung nicht zugunsten des Studierenden bewertet werden (dadurch ergibt sich eine Benotungsproblematik). Und nicht zuletzt unterscheiden sich Dummy-Datensätze wie z.B. Max Mustermann oder Lieschen Müller (je nach Fantasie der Studierenden) doch ganz erheblich von der realen Welt. Ein Test mit Dummy-Datensätzen sagt über die Qualität, Stabilität und vor allem Funktionalität einer Softwareanwendung relativ wenig aus.

- Kauf von professionell gesammelten Datensätzen – auch bei diesem Punkt sind die Erfolgsaussichten ganz gut, wenn es sich z.B. um Adressdaten unterschiedlicher Art handelt. In unserem speziellen Beispiel ist es aber nicht praktikabel, da wieder die o.g. Datenschutzbestimmungen greifen.

Fassen wir zusammen: Während der Projektarbeiten werden früher oder später in irgendeiner Form Massendaten benötigt. Diese sind i.d.R. wegen Datenschutzbestimmungen nicht verfügbar. Mit selbst erzeugten Testdatensätzen kann die Realität nicht hinreichend genau abgebildet werden.

Zur Eignungsprüfung der entwickelten Anwendung werden für die statistische Absicherung aber authentische Datensätze benötigt.

1.3 Massendaten aus der Natur oder Umwelt

Im Herbst 2006 entstand die Idee für ein Projekt ‘Wetterstation’. Die Natur, das Klima, das Wetter ändern sich ständig: Mittels Sensoren oder Messgeräten lassen sich verschiedene Parameter nach Bedarf in variablen Zeitintervallen erfassen und als Massendaten in ein DBS schreiben. Größte Vorteile waren, dass je nach Abfrageintervall die Menge der

Massendaten selber bestimmt (gesteuert) werden konnte und sofern die Messgeräte und Sensoren richtig geeicht waren, wurden authentische und reale Werte erzeugt.

Mit Hilfe des Förderverein Netzwerk Wirtschaft der Universität Lüneburg e.V. (der einen Teil der Kosten übernahm) konnte dann ziemlich schnell ein Regenmesser beschafft und auf dem Dach des Altbaus in Volgershall installiert werden. Dieser lieferte Messwerte in eine sogenannte CSV-Datei (Comma-Separated Values) im Filesystem des angeschlossenen Computers. Diese Textdatei wurde täglich mit Datum, Uhrzeit und Messwert erzeugt.

Ein Gespräch mit der Haustechnik ergab, dass auch eine Sensorik für die Gebäudesteuerung (Fenster, Beleuchtung, Heizung usw.) vorhanden wäre. Diese sei über einen sogenannten EIB (Europäischer Installationsbus) genormt ansprechbar, so dass noch mehr Massendaten über die Sensorik der Gebäudeautomation in das Projekt ‘Wetterstation’ mit eingebunden werden konnten (s. Tabelle 1).

Sensoren	Einheit der Meßwerte	Meßbereich
Dämmerung	Lux	0-255 Lux
Helligkeit	Lux	3.000 - 60.000 Lux
Regenmenge	mm NS	
Temperatur	Grad Celsius	-30 - 70 Grad C
Windgeschwindigkeit	m/s	0,5 - 50 m/s
Windrichtung	Grad	0 - 360 Grad

Tabelle 1: Sensoren für das Projekt ‘Wetterstation’ Stand Januar 2007

In einem ersten Ansatz wurde versucht, die Daten über einen Wetter-Service im Netzwerk abfragbar zu machen. Hierzu wurde eine SOAP-Anwendung auf Java Tomcat Application Server entwickelt (siehe http://rzserv2.uni-lueneburg.de:8080/weather/).

2 Das Projekt ‘Wetterstation’

Erste Projektgruppen im Sommersemester 2007 wurden gebildet mit der Idee eine Wetterseite für die Universitäts-Homepage zu erstellen. Ziel war die sinnvolle Visualisierung der von den Sensoren gelieferten Massendaten. Ferner sollten dabei die Probleme herausgearbeitet werden, die bei ständig neu anfallenden Massendaten auftreten können. Es gab zwei Gruppen mit drei und vier Studierenden, die den Wetter-Service über die Webschnittstelle abfragen und die gewonnenen Werte in einer dreidimensionalen Szene (3D) visualisieren sollten.

Eine Projektgruppe wählte zum Modellieren der 3D Szene das Tool Cinema 4D der Firma Maxxon, welches sich durch eine eigene Skriptsprache steuern ließ, während die andere Projektgruppe zum Modellieren die Software VUE der Firma Eon wählte (Steuerung mittels Python-Skripten), sowie zum Implementieren der Abfragelogik das graphische Programmiersystem LABVIEW der Firma National Instruments.

Die Studierenden bemängelten, dass für eine professionelle Wetterseite im Internet ja noch Werte wie relative Luftfeuchtigkeit und Luftdruck fehlten. Im Arbeitskreis Umwelt (AKU) der Universität erfuhren wir, dass es solche Sensoren auf dem Campus Scharnhorststraße bei der Umweltchemie (Prof. Dr. Ruck) geben solle. Nach einer ersten Kontaktaufnahme ermöglichten Mitarbeiter der Umweltchemie die Einbindung weiterer Messwerte mittels FTP-Zugriff.

Im Frühjahr 2008 konnte dann die erste Version der Wetterseite online gehen (http://wetter.uni-lueneburg.de). Für die Erfassung, Speicherung und Visualisierung der Massendaten wurden eine Vielzahl vernetzter Rechner, unterschiedliche Programmiersprachen und Protokolle eingesetzt (s. Abb. 1). Die Studierenden erfuhren dabei, wie wichtig die klare Definition von Schnittstellen für die Programmierung ist, um eine so komplexe Aufgabe von der Messkette bis hin zur Internet-Visualisierung zu meistern.

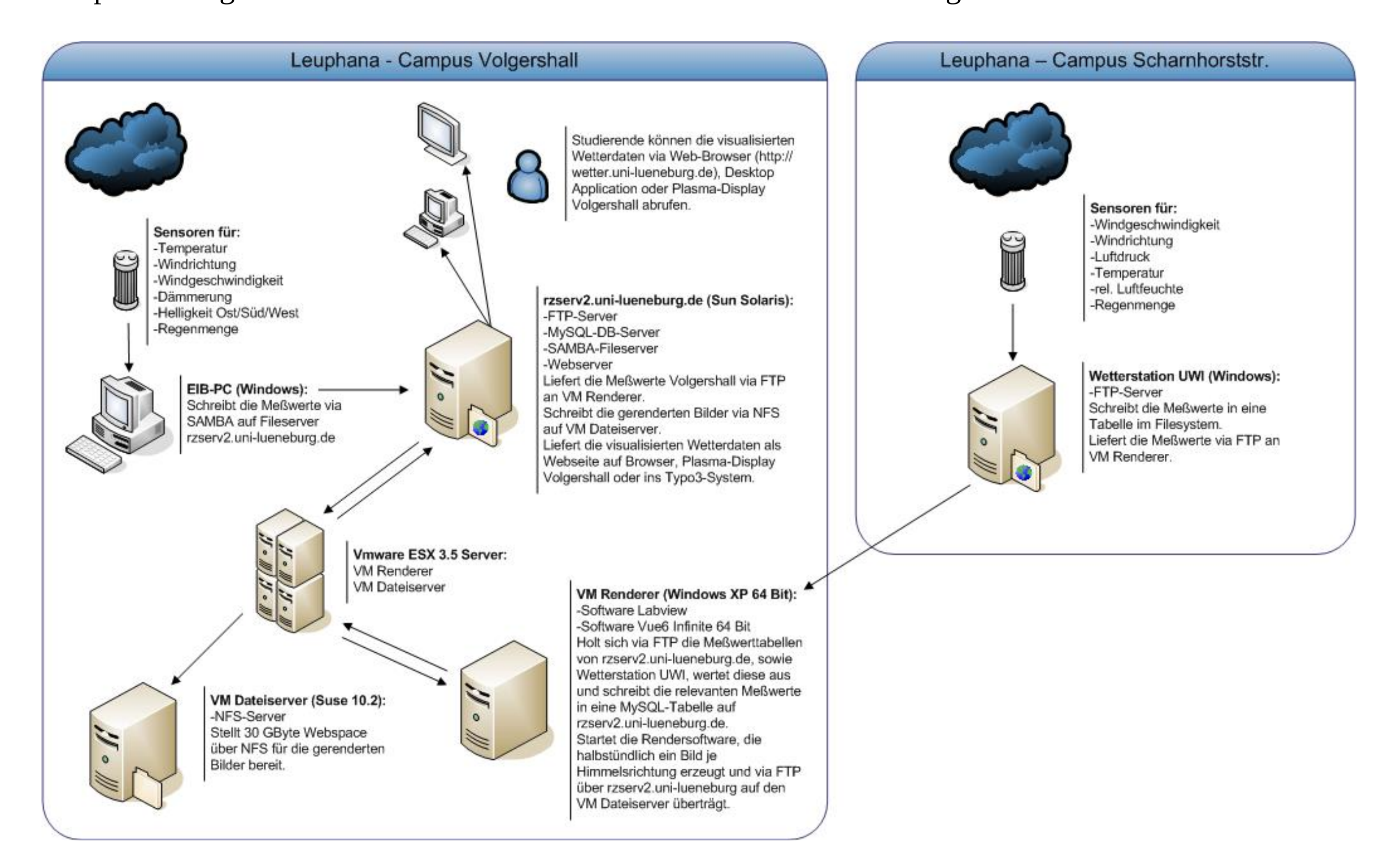


Abb. 1: Diagramm – Visualisierung von Massendaten am Beispiel ‘Wetterstation’

Nachteile der Projektarbeit ‘Wetterstation’:

- Durch lange Renderzeiten bekam man im Internet die Werte nur alle 30 Minuten angezeigt, also keine Live-Wetterdaten. Bsp.: Im Netz wird Regen angezeigt, der vor einer halben Stunde gefallen war, nun scheint die Sonne => dies ist besonders fatal, da das Internet seine Attraktivität aus der Aktualität von Informationen bezieht.

- Hoher Ressourcenverbrauch VMWare ESX-Server mit einer 64 Bit Windows XP virtuellen Maschine und 2 Prozessoren

- Gerenderte Bilder überlasten den Webserver rzserv2 (32 MB/Tag, 960 MB/Monat, 11,5 GB/Jahr). Zuerst wurde auf eine weitere virtuelle Maschine (Suse Linux) über NFS ausgewichen. Als der Fachbereich eine NAS-Lösung (Network Attached Storage) bekam, wurde für die Archivbilder dann über NFS ein Volumen von 100 GB freigestellt.

- Durch die Art ihrer Aufstellung lieferten einige Sensoren falsche Werte. Der Temperatursensor wurde durch Strahlungswärme des Hausdaches beeinflusst, der Windsensor durch Turbulenzen.

- Da das Hauptaugenmerk im ersten Ansatz auf der Visualisierung der Massendaten lag, entsprach die Datenhaltung nicht dem Stand der Technik. Es gab zu der Zeit eine kleine MySQL Datenbank, in der halbstündlich die Werte zu den gerenderten Bildern abgelegt waren, der Hauptanteil der Massendaten lag aber immer noch im Filesystem des FTP-Servers (als CSV-Dateien), war also für weitere Datenauswertungen nur mühselig zu gebrauchen.

3 Das Projekt ‘Klimadatenbank’

Um diese Nachteile zu beseitigen, wurde die Weiterführung des Projektes Wetterdaten im Wintersemester 2008/09 mit zwei Gruppen beschlossen. Dabei sollte sich eine Gruppe primär um die Probleme der Datenhaltung, Datenpersistierung und den Aufbau einer Klimadatenbank bemühen, während die andere Gruppe verschiedene Visualisierungsarten für das Internet in Echtzeit umsetzen sollte.

Die Projektziele wurden wie folgt definiert:

- Ausbau der Wetterdatenbank zur Klimadatenbank (mit berechneten Tagesdurchschnittswerten, gefühlter Temperatur und anderen relevanten Klimawerten).

- Optimierung der Sensoren durch Installation eines Kombinationssensors, der auf einem Fahnenmast auf dem Dach des Altbaus in Volgershall angebracht wurde. Gebäudeeffekte sind dadurch auszuschließen, und es wurden alle wetterrelevanten Parameter in einem Durchgang gemessen und archiviert.

- Installation eines zusätzlichen Sensors, um den CO_2-Gehalt in der Luft zu bestimmen.

- Archivierung von Live-Messwerten (d.h. mindestens minütlich werden die Werte abgefragt, in die Klimadatenbank geschrieben und visualisiert).

- Verschiedene Visualisierungsarten, z.B. animierter 2D-Flash-Film mit Actionscript, sowie ein Archiv mit frei wählbaren Diagramm-Ansichten über verschiedene Zeiträume (s. Abb. 2).

- Versuch einer Serverkonsolidierung, d.h. möglichst wenig Technik, um ans Ziel zu kommen (dadurch Minimierung des Energieverbrauchs).

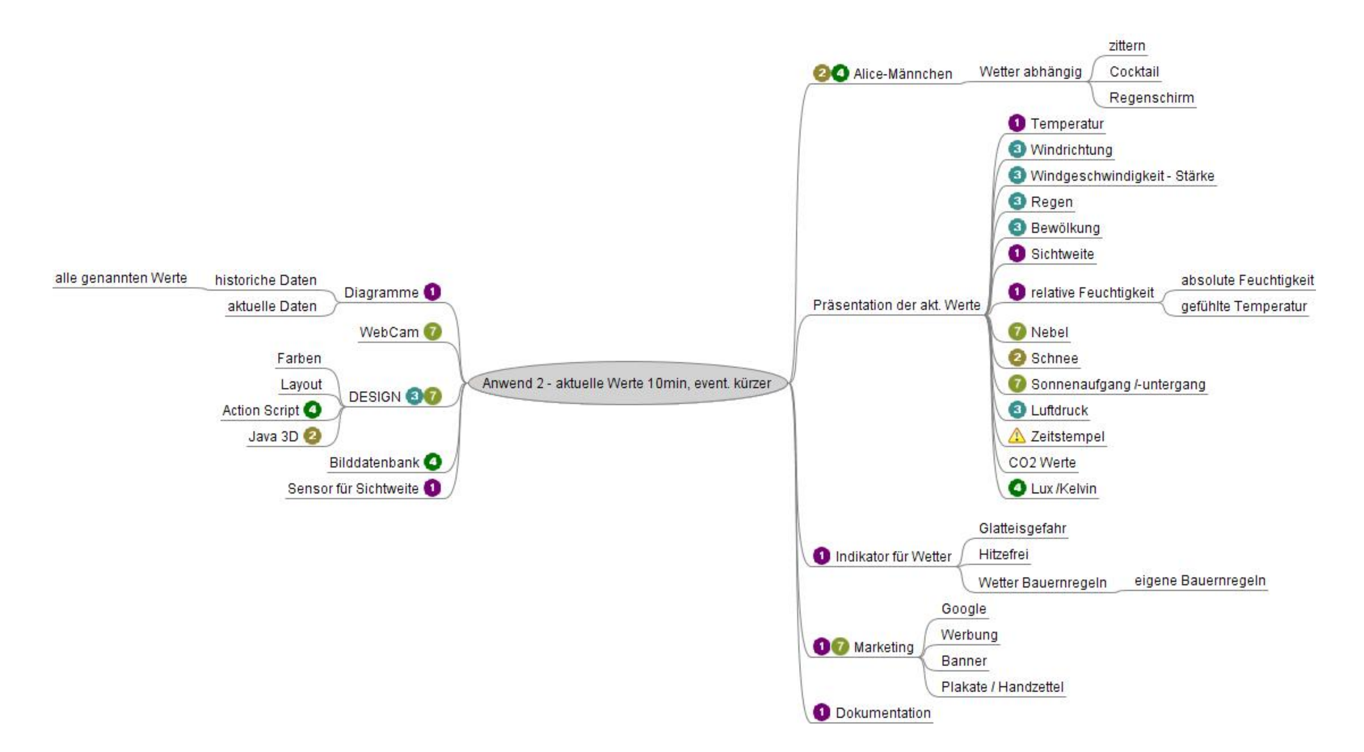


Abb. 2: Mindmap der Visualisierungsgruppe [4]

Projektende für die Gruppen Klimadatenbank/Datenhaltung und Visualisierung war dann im Frühjahr 2009.

Erreichte Ziele:

- Es wurde ein Java-Programm (Daten-Logger) erstellt, das die Werte von den verschiedenen Sensoren minütlich liest, daraus weitere relevante Werte berechnet (s. Abb. 4) und in die Klimadatenbank (s. Abb. 3) schreibt (Implementierung als Thread).

- Eine zusätzliche Visualisierung mittels wechselndem Live-Bild per Webcam (vom Dach Volgershall) wurde auf der Internetseite (http://www.leuphana.de/wetter) implementiert.

- Es wurde eine zweidimensionale Visualisierung mittels Flash-Film auf der Internetseite eingefügt.

- Durch die Montage der Sensorik auf einem Fahnenmast ca. 9 m über dem Dach des Altbaus in Volgershall erhielt man korrekte Messwerte.

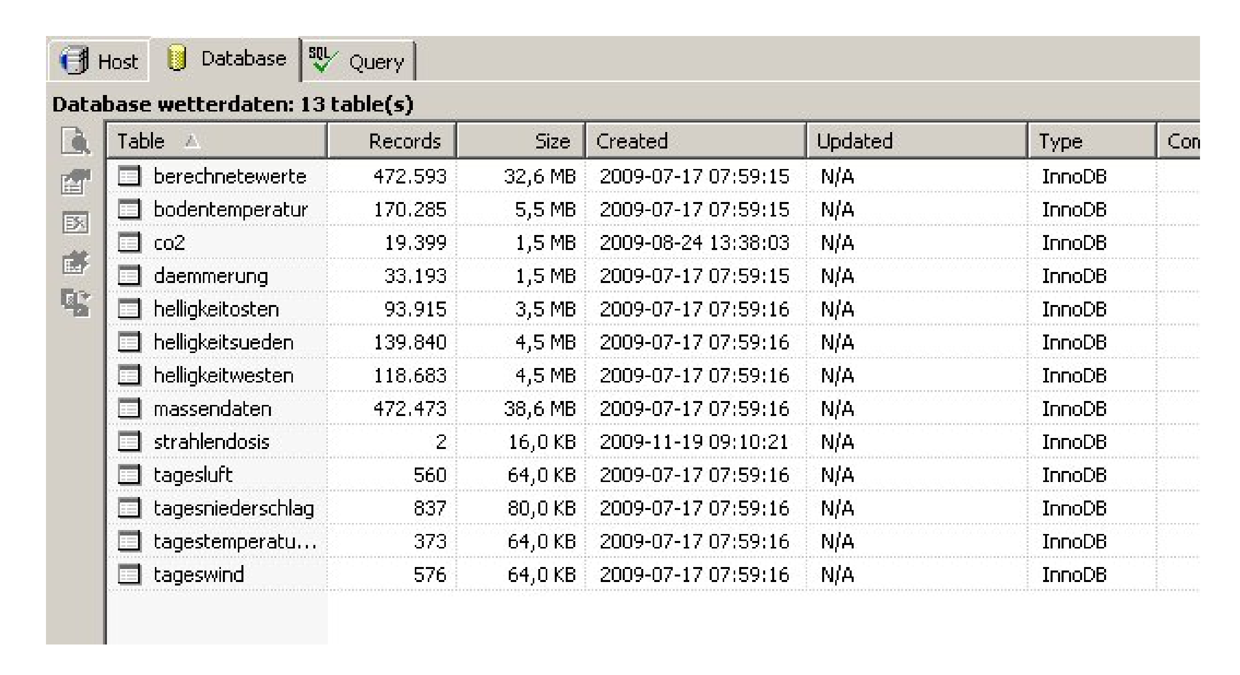
- Weniger aktive Systeme werden benötigt (Energieeinsparung).

Nicht erreichte Ziele:

- Der CO2-Meßsensor wurde zwar installiert, aber programmiertechnisch noch nicht eingebunden. Die Messwerte waren daher zu der Zeit nur lokal und nicht im Netzwerk verfügbar.

- Die Visualisierung über Diagramme und frei wählbare Zeiträume war nicht funktionsfähig, bzw. arbeitete mit falschen Werten. Daher wurde vorläufig das alte VUE Archiv auf der Internetseite eingebunden.

Im Sommer 2009 wurde ein Programm erstellt, das nun auch die CO2-Werte von der Meßsonde in die Klimadatenbank schreibt. Alle Visualisierungen wurden angepasst.



Host | Database | Query

Database wetterdaten: 13 table(s)

Table	Records	Size	Created	Updated	Type	Con
berechnetewerte	472.593	32,6 MB	2009-07-17 07:59:15	N/A	InnoDB	
bodentemperatur	170.285	5,5 MB	2009-07-17 07:59:15	N/A	InnoDB	
co2	19.399	1,5 MB	2009-08-24 13:38:03	N/A	InnoDB	
daemmerung	33.193	1,5 MB	2009-07-17 07:59:15	N/A	InnoDB	
helligkeitosten	93.915	3,5 MB	2009-07-17 07:59:16	N/A	InnoDB	
helligkeitsueden	139.840	4,5 MB	2009-07-17 07:59:16	N/A	InnoDB	
helligkeitwesten	118.683	4,5 MB	2009-07-17 07:59:16	N/A	InnoDB	
massendaten	472.473	38,6 MB	2009-07-17 07:59:16	N/A	InnoDB	
strahlendosis	2	16,0 KB	2009-11-19 09:10:21	N/A	InnoDB	
tagesluft	560	64,0 KB	2009-07-17 07:59:16	N/A	InnoDB	
tagesniederschlag	837	80,0 KB	2009-07-17 07:59:16	N/A	InnoDB	
tagestemperatu...	373	64,0 KB	2009-07-17 07:59:16	N/A	InnoDB	
tageswind	576	64,0 KB	2009-07-17 07:59:16	N/A	InnoDB	

Abb. 3: Tabellen der Klimadatenbank (Werte vom 15.1.2010)

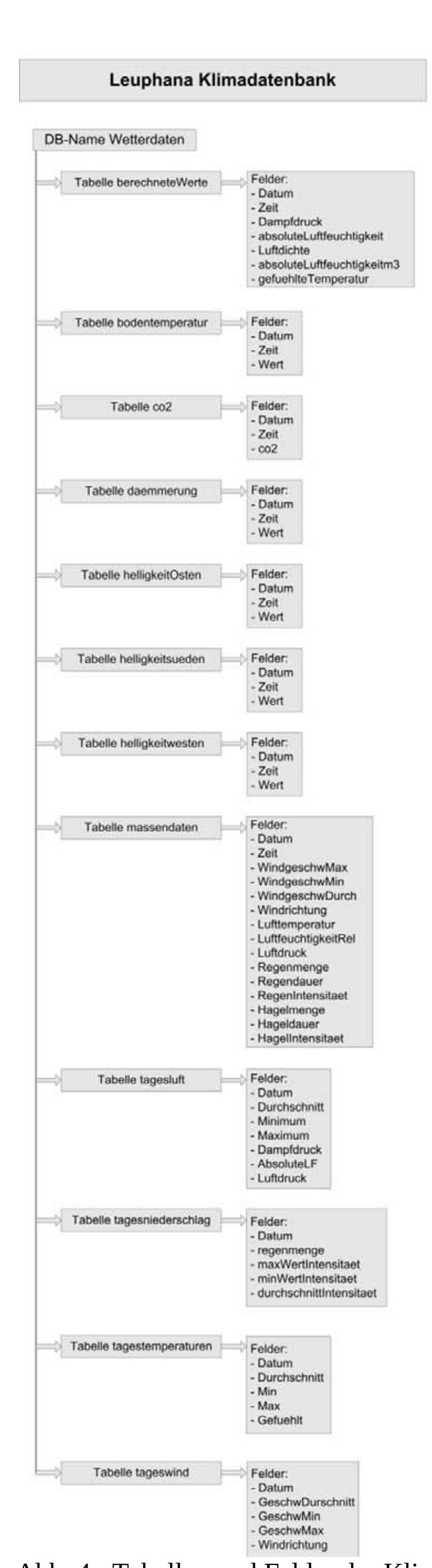




Abb. 4: Tabellen und Felder der Klimadatenbank

4 Das Projekt ‘Klimadatenbank 2 / Klimaarchiv’

Im Wintersemester 2009/10 wurde das Projekt Klimadatenbank mit drei Projektgruppen fortgeführt. Als Projektziele wurden die Einbindung zusätzlicher Sensorik für Umweltdaten (hier speziell ein Gamma-Strahlen Dosismessgerät), sowie die Ablösung der alten Archivfunktion im Internet und die Visualisierung über Diagramme und frei wählbare Zeiträume angestrebt. Voraussichtliches Projektende Frühjahr 2010.

5 Anhang - Bilder der verschiedenen Visualisierungsarten

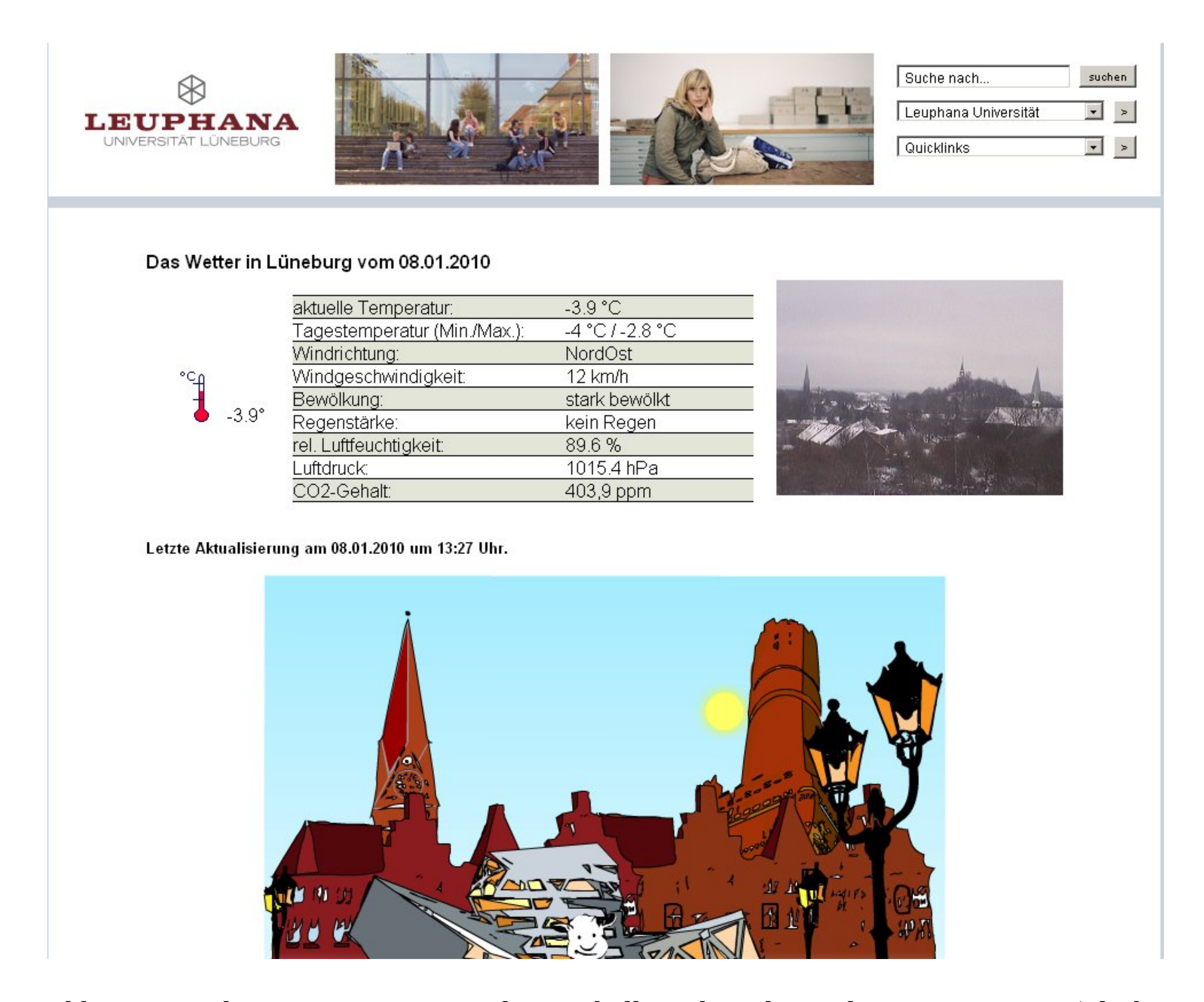


Abb. 5: Visualisierung von Massendaten tabellarisch und mittels 2D Animation (Flash-Film) unter http://www.leuphana.de/wetter

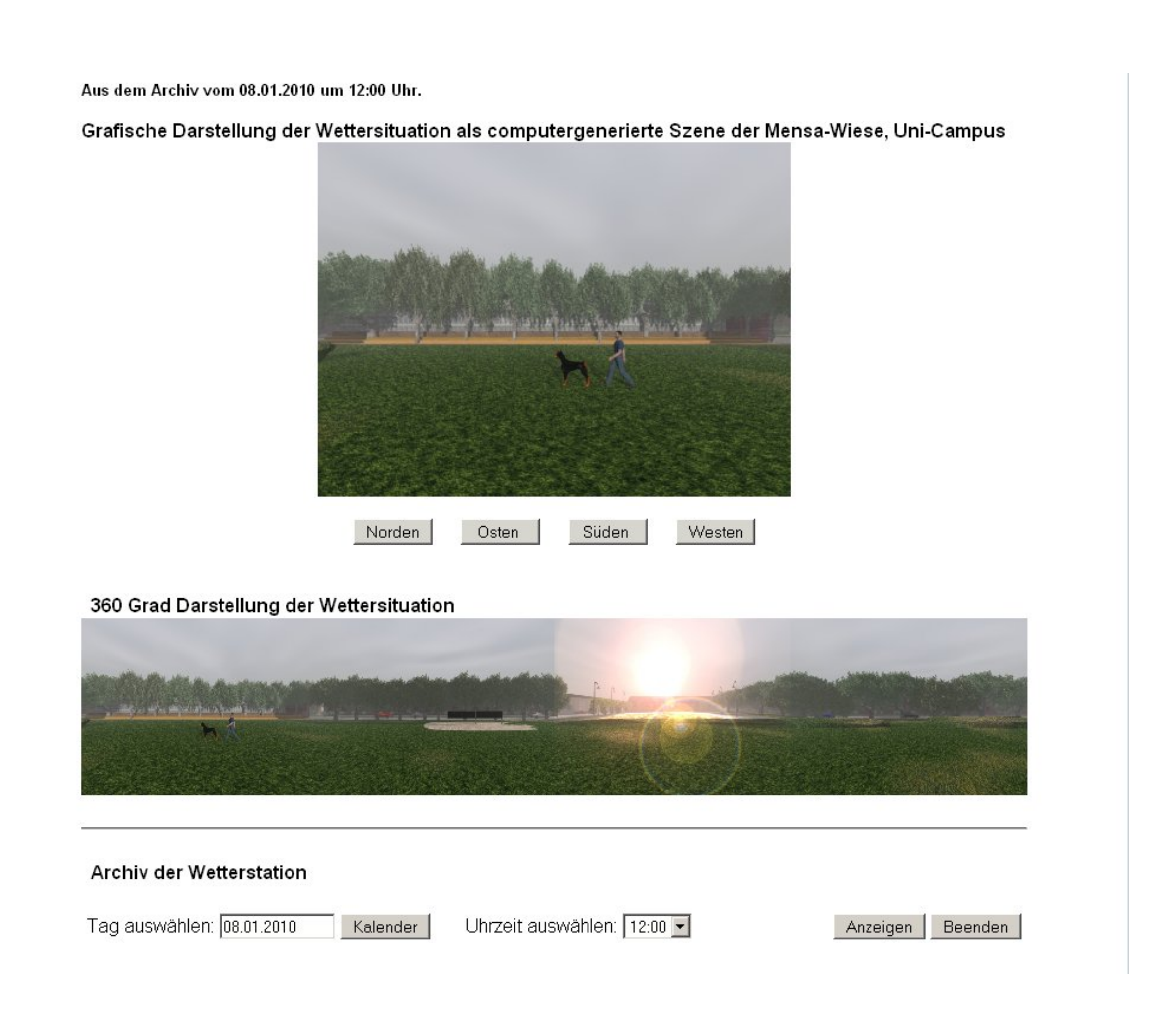


Abb. 6: Computergenerierte Wettersituation aus dem Wetterarchiv

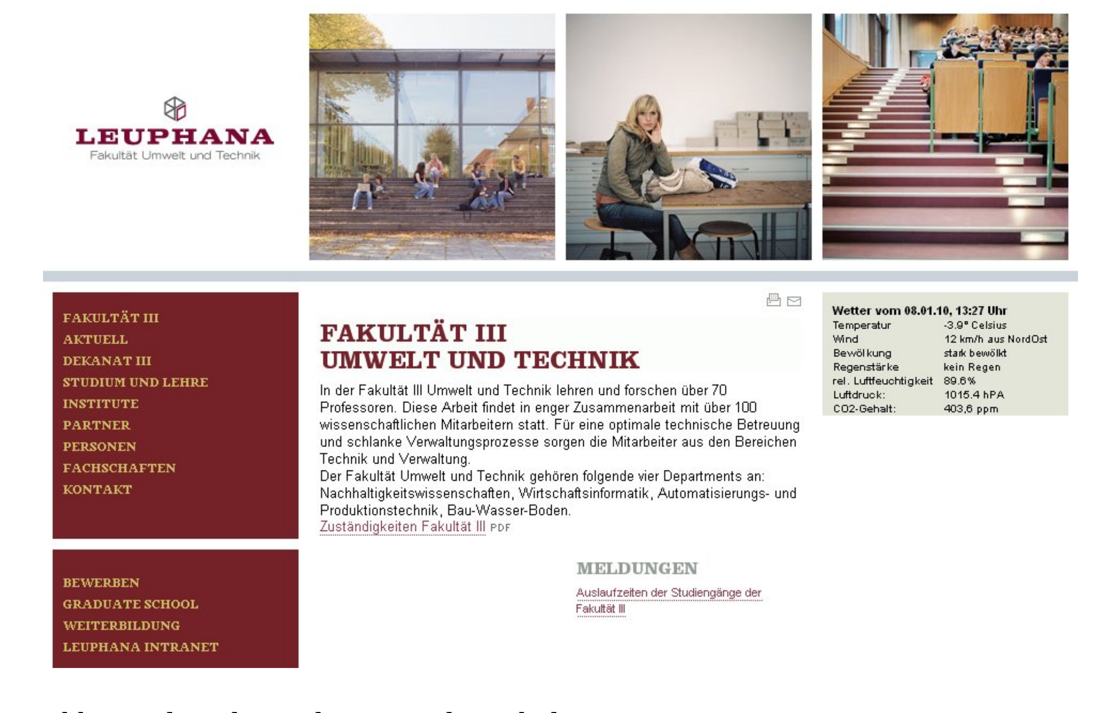


Abb. 7: Klimadaten als Teaser der Fakultätsseite im Content-Management-System Typo3

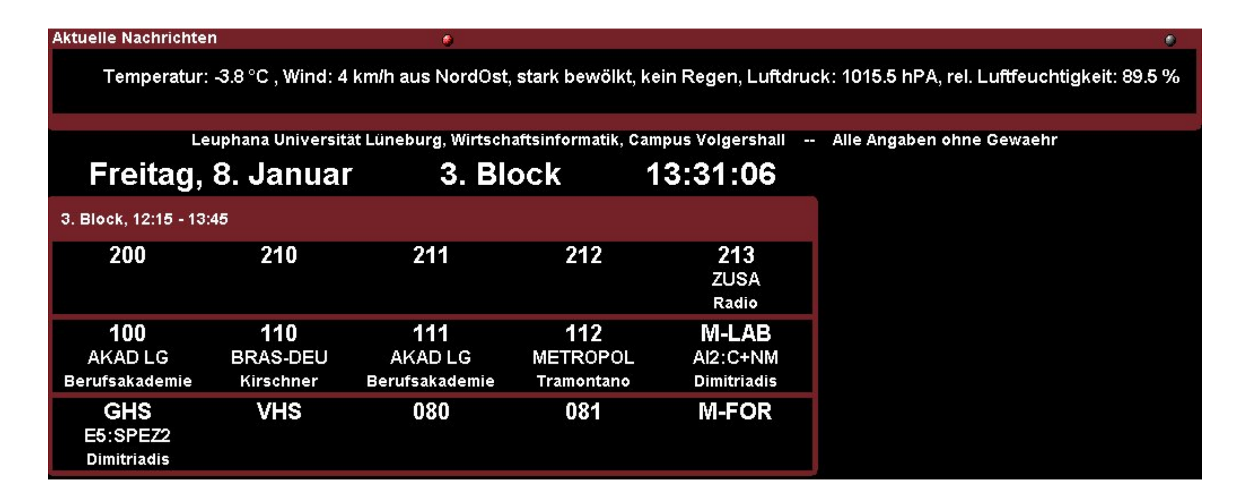


Abb. 8: Klimadaten auf dem Plasma-Bildschirm Stundenplan in der Eingangshalle Volgershall/Altbau

Klimadaten vom 08.01.2010 um 13:22 Uhr.	
aktuelle Temperatur:	-3.9° Celsius
Tagestemp. Min./Max.:	-3.9° Celsius / -2.8° Celsius
Windrichtung:	NordOst
Windgeschwindigkeit:	14 km/h
Bewölkung:	stark bewölkt
Regenstärke:	kein Regen
rel. Luftfeuchtigkeit:	89.5%
Luftdruck:	1015.3 hPa
CO2-Gehalt:	405,6 ppm

Abb. 9: Klimadaten als Adobe Air Applikation für den Desktop

6 Literaturverzeichnis

[1] Prof. Dr. Bonin , Wirtschaftsinformatik … ein weites Feld, Juli 2008

[2] Internet, mystudy gesehen am14.1.10

[3] Internet, Wikipedia, die freie Enzyklopädie, http://de.wikipedia.org/wiki/Datenbank gesehen am 19.1.10

[4] Müller, Golubchykov, Hoffmann, Cam Duy Nguyen, Jensch, Anwendungsentwicklung 2 - Visualisierung der Klimasensoren – Hausarbeit vom 1.4.2009, Version 1.0

In der Reihe FINAL sind bisher erschienen:

1. Jahrgang 1991:

1. Hinrich E. G. Bonin; Softwaretechnik, Heft 1, 1991 (ersetzt durch Heft 2, 1992).
2. Hinrich E. G. Bonin (Herausgeber); Konturen der Verwaltungsinformatik, Heft 2, 1991 (überarbeitet und erschienen im Wissenschaftsverlag, Bibliographisches Institut & F. A. Brockhaus AG, Mannheim 1992, ISBN 3-411-15671-6).

2. Jahrgang 1992:

1. Hinrich E. G. Bonin; Produktionshilfen zur Softwaretechnik --- Computer-Aided Software Engineering --- CASE, Materialien zum Seminar 1992, Heft 1, 1992.
2. Hinrich E. G. Bonin; Arbeitstechniken für die Softwareentwicklung, Heft 2, 1992 (3. überarbeitete Auflage Februar 1994), PDF-Format.
3. Hinrich E. G. Bonin; Object-Orientedness --- a New Boxologie, Heft 3, 1992.
4. Hinrich E. G. Bonin; Objekt-orientierte Analyse, Entwurf und Programmierung, Materialien zum Seminar 1992, Heft 4, 1992.
5. Hinrich E. G. Bonin; Kooperative Produktion von Dokumenten, Materialien zum Seminar 1992, Heft 5, 1992.

3. Jahrgang 1993:

1. Hinrich E. G. Bonin; Systems Engineering in Public Administration, Proceedings IFIP TC8/ WG8.5: Governmental and Municipal Information Systems, March 3--5, 1993, Lüneburg, Heft 1, 1993 (überarbeitet und erschienen bei North-Holland, IFIP Transactions A-36, ISSN 0926-5473).
2. Antje Binder, Ralf Linhart, Jürgen Schultz, Frank Sperschneider, Thomas True, Bernd Willenbockel; COTEXT --- ein Prototyp für die kooperative Produktion von Dokumenten, 19. März 1993, Heft 2, 1993.
3. Gareth Harries; An Introduction to Artificial Intelligence, April 1993, Heft 3, 1993.
4. Jens Benecke, Jürgen Grothmann, Mark Hilmer, Manfred Hölzen, Heiko Köster, Peter Mattfeld, Andre Peters, Harald Weiss; ConFusion --- Das Produkt des AWÖ-Projektes 1992/93, 1. August 1993, Heft 4, 1993.
5. Hinrich E. G. Bonin; The Joy of Computer Science --- Skript zur Vorlesung EDV ---, September 1993, Heft 5, 1993 (4. ergänzte Auflage März 1995).
6. Hans-Joachim Blanke; UNIX to UNIX Copy --- Interactive application for installation and configuration of UUCP ---, Oktober 1993, Heft 6, 1993.

4. Jahrgang 1994:

1. Andre Peters, Harald Weiss; COMO 1.0 --- Programmierumgebung für die Sprache COBOL --- Benutzerhandbuch, Februar 1994, Heft 1, 1994.
2. Manfred Hölzen; UNIX-Mail --- Schnelleinstieg und Handbuch ---, März 1994, Heft 2, 1994.
3. Norbert Kröger, Roland Seen; EBrain --- Documentation of the 1994 AWÖ-Project Prototype ---, June 11, 1994, Heft 3, 1994.
4. Dirk Mayer, Rainer Saalfeld; ADLATUS --- Documentation of the 1994 AWÖ-Project Prototype -- -, July 26, 1994, Heft 4, 1994.
5. Ulrich Hoffmann; Datenverarbeitungssystem 1, September 1994, Heft 5, 1994. (2. überarbeitete Auflage Dezember 1994).
6. Karl Goede; EDV-gestützte Kommunikation und Hochschulorganisation, Oktober 1994, Heft 6 (Teil 1), 1994.
7. Ulrich Hoffmann; Zur Situation der Informatik, Oktober 1994, Heft 6 (Teil 2), 1994.

5. Jahrgang 1995:

1. Horst Meyer-Wachsmuth; Systemprogrammierung 1, Januar 1995, Heft 1, 1995.
2. Ulrich Hoffmann; Datenverarbeitungssystem 2, Februar 1995, Heft 2, 1995.
3. Michael Guder / Kersten Kalischefski / Jörg Meier / Ralf Stöver / Cheikh Zeine; OFFICE-LINK --- Das Produkt des AWÖ-Projektes 1994/95, März 1995, Heft 3, 1995.
4. Dieter Riebesehl; Lineare Optimierung und Operations Research, März 1995, Heft 4, 1995.
5. Jürgen Mattern / Mark Hilmer; Sicherheitsrahmen einer UTM-Anwendung, April 1995, Heft 5, 1995.
6. Hinrich E. G. Bonin; Publizieren im World-Wide Web --- HyperText Markup Language und die Kunst der Programmierung ---, Mai 1995, Heft 6, 1995.
7. Dieter Riebesehl; Einführung in Grundlagen der theoretischen Informatik, Juli 1995, Heft 7, 1995.
8. Jürgen Jacobs; Anwendungsprogrammierung mit Embedded-SQL, August 1995, Heft 8, 1995.
9. Ulrich Hoffmann; Systemnahe Programmierung, September 1995, Heft 9, 1995 (ersetzt durch Heft 1, 1999).
10. Klaus Lindner; Neuere statistische Ergebnisse, Dezember 1995, Heft 10, 1995.

6. Jahrgang 1996:

1. Jürgen Jacobs / Dieter Riebesehl; Computergestütztes Repetitorium der Elementarmathematik, Februar 1996, Heft 1, 1996.
2. Hinrich E. G. Bonin; "Schlanker Staat" & Informatik, März 1996, Heft 2, 1996.
3. Jürgen Jacobs; Datenmodellierung mit dem Entity-Relationship-Ansatz, Mai 1996, Heft 3, 1996.
4. Ulrich Hoffmann; Systemnahe Programmierung, (2. überarbeitete Auflage von Heft 9, 1995), September 1996, Heft 4, 1996 (ersetzt durch Heft 1, 1999).
5. Dieter Riebesehl; Prolog und relationale Datenbanken als Grundlagen zur Implementierung einer NF2-Datenbank (Sommer 1995), November 1996, Heft 5, 1996.

7. Jahrgang 1997:

1. Jan Binge, Hinrich E. G. Bonin, Volker Neumann, Ingo Stadtsholte, Jürgen Utz; Intranet-/Internet- Technologie für die Öffentliche Verwaltung --- Das AWÖ-Projekt im WS96/97 --- (Anwendungen in der Öffentlichen Verwaltung), Februar 1997, Heft 1, 1997.
2. Hinrich E. G. Bonin; Auswirkungen des Java-Konzeptes für Verwaltungen, FTVI'97, Oktober 1997, Heft 2, 1997.

8. Jahrgang 1998:

1. Hinrich E. G. Bonin; Der Java-Coach, Heft 1, Oktober 1998, (CD-ROM, PDF-Format; aktuelle Fassung).
2. Hinrich E. G. Bonin (Hrsg.); Anwendungsentwicklung WS 1997/98 --- Programmierbeispiele in COBOL & Java mit Oracle, Dokumentation in HTML und tcl/tk, September 1998, Heft 2, 1998 (CD-ROM).
3. Hinrich E. G. Bonin (Hrsg); Anwendungsentwicklung SS 1998 --- Innovator, SNiFF+, Java, Tools, Oktober 1998, Heft 3, 1998 (CD-ROM).
4. Hinrich E. G. Bonin (Hrsg); Anwendungsentwicklung WS 1998 --- Innovator, SNiFF+, Java, Mail und andere Tools, November 1998, Heft 4, 1998 (CD-ROM).
5. Hinrich E. G. Bonin; Persistente Objekte --- Der Elchtest für ein Java-Programm, Dezember 1998, Heft 5, 1998 (CD-ROM).

9. Jahrgang 1999:

1. Ulrich Hoffmann; Systemnahe Programmierung (3. überarbeitete Auflage von Heft 9, 1995), Juli 1999, Heft 1, 1999 (CD-ROM und Papierform), Postscript-Format, zip-Postscript-Format, PDF-Format und zip-PDF-Format.

10. Jahrgang 2000:

1. Hinrich E. G. Bonin; Citizen Relationship Management, September 2000, Heft 1, 2000 (CD-ROM und Papierform), PDF-Format.
2. Hinrich E. G. Bonin; WI>DATA --- Eine Einführung in die Wirtschaftsinformatik auf der Basis der Web_Technologie, September 2000, Heft 2, 2000 (CD-ROM und Papierform), PDF-Format.
3. Ulrich Hoffmann; Angewandte Komplexitätstheorie, November 2000, Heft 3, 2000 (CD-ROM und Papierform), PDF-Format.
4. Hinrich E. G. Bonin; Der kleine XMLer, Dezember 2000, Heft 4, 2000 (CD-ROM und Papierform), PDF-Format, aktuelle Fassung.

11. Jahrgang 2001:

1. Hinrich E. G. Bonin (Hrsg.): 4. SAP-Anwenderforum der FHNON, März 2001, (CD-ROM und Papierform), Downloads & Videos.
2. J. Jacobs / G. Weinrich; Bonitätsklassifikation kleiner Unternehmen mit multivariater linear Diskriminanzanalyse und Neuronalen Netzen; Mai 2001, Heft 2, 2001, (CD-ROM und Papierform), PDF-Format und MS Word DOC-Format
3. K. Lindner; Simultanttestprozedur für globale Nullhypothesen bei beliebiger Abhängigkeitsstruktur der Einzeltests, September 2001, Heft 3, 2001 (CD-ROM und Papierform).

12. Jahrgang 2002:

1. Hinrich E. G. Bonin: Aspect-Oriented Software Development. März 2002, Heft 1, 2002 (CD-ROM und Papierform), PDF-Format.
2. Hinrich E. G. Bonin: WAP & WML --- Das Projekt Jagdzeit ---. April 2002, Heft 2, 2002 (CD-ROM und Papierform), PDF-Format.
3. Ulrich Hoffmann: Ausgewählte Kapitel der Theoretischen Informatik (CD-ROM und Papierform), PDF-Format.
4. Jürgen Jacobs / Dieter Riebesehl; Computergestütztes Repetitorium der Elementarmathematik, September 2002, Heft 4, 2002 (CD-ROM und Papierform), PDF-Format.
5. Verschiedene Referenten; 3. Praxisforum "Systemintegration", 18.10.2002, Oktober 2002, Heft 5, 2002 (CD-ROM und Papierform), Praxisforum.html (Web-Site).

13. Jahrgang 2003:

1. Ulrich Hoffmann; Ausgewählte Kapitel der Theoretischen Informatik; Heft 1, 2003, (CD-ROM und Papierform) PDF-Format.
2. Dieter Riebesehl; Mathematik 1, Heft 2, 2003, (CD-ROM und Papierform) PDF-Format.
3. Ulrich Hoffmann; Mathematik 1, Heft 3, 2003, (CD-ROM und Papierform) PDF-Format und Übungen.
4. Verschiedene Autoren; Zukunft von Verwaltung und Informatik, Festschrift für Heinrich Reinermann, Heft 4, 2003, (CD-ROM und Papierform) PDF-Format.

14. Jahrgang 2004:

1. Jürgen Jacobs; Multilayer Neural Networks; Heft 1, 2004, (CD-ROM und Papierform) PDF-Format.

15. Jahrgang 2005:

1. Ulrich Hoffmann; Mathematik für Wirtschaftsinformatiker; Heft 1, 2005, (CD-ROM und Papierform) PDF-Format.
2. Ulrich Hoffmann; Übungen & Lösungen zur Mathematik für Wirtschaftsinformatiker; Heft 1, 2005, (CD-ROM und Papierform) PDF-Format.
3. Ulrich Hoffmann; Datenstrukturen & Algorithmen; Heft 2, 2005, (CD-ROM und Papierform) PDF-Format.

16. Jahrgang 2006:

1. Hinrich E. G. Bonin; Systemanalyse für Softwaresysteme; Heft 1, August 2006, (CD-ROM und Papierform) PDF-Format.
2. Hinrich E. G. Bonin; Faszination Programmierung; Heft 2, August 2006, (CD-ROM und Papierform) PDF-Format.
3. Dieter Riebesehl; Strukturanalogien in Datenmodellen, Heft 3, Dezember 2006, (CD-ROM und Papierform) PDF-Format.

17. Jahrgang 2007:

1. Ulrich Hoffmann; Ausgewählte Kapitel der Theoretischen Informatik; Heft 1, August 2007, (CD-ROM und Papierform) PDF-Format.
2. Ulrich Hoffmann; Mathematik für Wirtschaftsinformatiker und Informatiker; Heft 2, August 2007, (CD-ROM und Papierform) PDF-Format.
3. Hinrich E. G. Bonin; Der Java-Coach, Heft 3, September 2007, (CD-ROM und Papierform) PDF-Format.
4. Jürgen Jacobs; Dichteprognose autoregressiver Zeitreihen, Heft 4, September 2007, (CD-ROM und Papierform) PDF-Format.

18. Jahrgang 2008:

1. Verschiedene Autoren; Festschrift für Prof. Dr. Meyer-Wachsmuth; Heft 1, Juli 2008, (CD-ROM und Papierform) PDF-Format.
2. Ulrich Hoffmann; Ausgewählte Kapitel der Theoretischen Informatik; Heft 2, Dezember 2008, (CD-ROM und Papierform) PDF-Format.

19. Jahrgang 2009:

1. Verschiedene Autoren; Festschrift für Prof. Dr. Goede; Heft 1, August 2009, (CD-ROM und Papierform) PDF-Format.

20. Jahrgang 2010:

1. Hinrich E. G. Bonin; Konstrukte, Konstruktionen, Konstruktionsempfehlungen – Programmieren in LISP; Heft 1, März 2010, (CD-ROM und Papierform) PDF-Format.
2. Verschiedene Autoren; Festschrift für Prof. Dr. Bonin; Heft 2, April 2010, (CD-ROM und Papierform) PDF-Format.

Herausgeber für diese Ausgabe:
Prof. Dr. Horst Meyer-Wachsmuth
Leuphana Universität Lüneburg, Volgershall 1, D-21339 Lüneburg, Germany
email: mw@uni.leuphana.de

Lektorat für diese Ausgabe:
Maja Irmhild Schütte-Hoof MA
Lektorin für Deutsch im Fremdsprachenzentrum der Leuphana Universität Lüneburg

Verlag:
Eigenverlag (Fotographische Vervielfältigung), Leuphana Universität Lüneburg
(vormals Fachhochschule Nordostniedersachsen)

Erscheinungsweise:
ca. 4 Hefte pro Jahr.
Für unverlangt eingesendete Manuskripte wird nicht gehaftet. Sie sind aber willkommen.

Digitales FInAL-Archiv:
http://www.leuphana.de/institute/iwi/final.html





Prof. Dr. Hinrich E. G. Bonin